Abstract of "Non-Redundant Clustering" by David Gondek, Ph.D., Brown University, May 2005.

Data mining and knowledge discovery attempt to reveal concepts, patterns, relationships, and structures of interest in data. Typically, data may have many such structures. Most existing data mining techniques allow the user little say in which structure will be returned from the search. Those techniques which do allow the user control over the search typically require supervised information in the form of knowledge about a target solution. In the spirit of exploratory data mining, we consider the setting where the user does not have information about a target solution. Instead we suppose the user can provide information about solutions which are *not desired*. These undesired solutions may be previously obtained from data mining algorithms, or they may be known to the user a priori. The goal is then to discover novel structure in the dataset which is not redundant with respect to the known structure. Techniques should guide the search away from this known structure and towards novel, interesting structures. We describe and formally define the task of non-redundant clustering. Three different algorithmic approaches are derived for non-redundant clustering. Their performance is experimentally evaluated on data sets containing multiple clusterings. We explore how these techniques may be extended to systematically enumerate clusterings in a data set. Finally, we also investigate whether non-redundant approaches may be incorporated to enhance state-of-the-art supervised techniques.

Non-Redundant Clustering

by

David Gondek

B. S., Dartmouth College, 1998

Sc. M., Brown University, 2000

A dissertation submitted in partial fulfillment of the

requirements for the Degree of Doctor of Philosophy

in the Department of Computer Science at Brown University

Providence, Rhode Island

May 2005

This dissertation by David Gondek is accepted in its present form by

the Department of Computer Science as satisfying the dissertation requirement

for the degree of Doctor of Philosophy.

Date _____ _____

Thomas Hofmann, Director

Recommended to the Graduate Council

Date _____ _____

Amy Greenwald, Reader

Date _____ _____

Shivakumar Vaithyanathan, Reader

Approved by the Graduate Council

Date _____ _____

Karen Newman
Dean of the Graduate School and Research

Acknowledgements

I would like to thank my advisor, Thomas Hofmann, for his invaluable help in suggesting effective solutions and directions for research. This work would not be possible without Shiv Vaithyanathan who first formulated the problem to me. Thanks also to Ashutosh Garg for many helpful and insightful discussions.

Contents

⋆ Parts of this dissertation have been previously published in [Vaithyanathan and Gondek, 2002a] co-written with Shivakumar Vaithyanathan, [Gondek and Hofmann, 2003, Gondek and Hofmann, 2004, Gondek and Hofmann, 2005] co-written with Thomas Hofmann, and [Gondek et al., 2005] co-written with Shivakumar Vaithyanathan and Ashutosh Garg.

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Exploratory Data Mining

Data mining and knowledge discovery attempt to reveal concepts, patterns, relationships, and structures of interest in data. In practice, however, these techniques allow little say over what structures are or are not of interest. As pointed out in work such as [Ng et al., 1998], existing data mining techniques are often designed as black boxes and do not necessarily support user interactivity and control which would allow for true exploratory data mining. To support interactivity, techniques should allow users to interactively refine or modify queries based on results of previous queries. To support user control, techniques should allow for the user to provide prior knowledge in order to guide the search, both toward desired solutions and away from known or undesired solutions.

It is this latter capability, to guide search away from known solutions, which is the focus of this work. We will propose an approach to provide interactive exploration and incorporate such prior knowledge for the task of clustering. In particular, we define the setting of *non-redundant clustering*. This setting differs crucially from existing interactive techniques which for the most part require knowledge of the target solution. In non-redundant clustering, by contrast, there is instead knowledge of undesired solutions. For instance, there may be a known classification scheme which

is not interesting to the user. The goal then is to discover a novel clustering which augments, not restates, this known classification scheme. This novel structure should be "orthogonal" to the known scheme. The result is a flexible system which can provide interactive and truly exploratory sessions.

1.1.1 Overview

In this chapter, we discuss the problem and the settings in which the techniques subsequently presented may be applied. In Chapter 3, we review existing clustering techniques. We discuss both those techniques upon which our algorithms have been designed as well as those which are used by existing semi-supervised algorithms. Following the related work, in Chapters 5, 6, and 7 we present our contributions which consist of three different algorithmic approaches to the problem of non-redundant clustering. In Chapter 8, we experimentally evaluate and compare these approaches. Chapter 9 discusses several extensions of the methods to related problems. Finally, in Chapter 10 we summarize and discuss potential directions for future work.

1.2 Non-Redundant Clustering

In the general framework, we are searching for a structure C which is non-redundant with respect to known structure Z. This relationship will be more rigorously defined in Section 4.2; we currently consider the forms Z may take and the settings in which it may be applied. There are two general forms of known structure Z:

- **Known structure** ($Z = C'$) A known structure, C' is given. For clustering, this would be a known categorization scheme. C' can be an existing hand-labeled structure or the result of a previous algorithm.
- **Noise features** ($Z = Y^-$) A set of "noise" features, Y^-, desired to be irrelevant are given. In the case of clustering, a solution is desired where the specified features Y^- are conditionally independent of the clustering C.

Known Structure

In the first case, a known categorization is given as known structure. This categorization may be the result of hand-labeled classification, it may be derived from the collection process (e.g. source websites of webpages), or it could be a classification obtained from supervised or unsupervised learning algorithms. This last form is particularly intriguing as it suggests a program of successive non-redundant clusterings. In particular, the known clusterings are given as known structure and the goal is to find a clustering which is novel with respect to them. This allows for a true exploratory data mining framework, where the results of previous searches may be used to guide new searches toward novel structure.

Supervised information may be available in addition to known structure. In an exploratory setting, it may be reasonable to expect the analyst to provide a small number of supervised examples (known as *semi-supervised* clustering.) This leads to the important question of whether a combined approach is possible, where non-redundant clustering algorithms are combined with existing semi-supervised methods in order to enhance the performance of these semi-supervised methods.

Noise features

In the second case, a set of features are given as irrelevant, or "noise" features. A solution C is desired in the context of which, the features Z appear as noise. Note that this notion of noise features differs from that presented in work such as [Vaithyanathan and Dom, 1999b, Law et al., 2002]. In these works, noise features are defined as irrelevant features resulting from the learned structure. Instead our use of the term *noise features* refers to features *desired* to appear noisy. So in any solution obtained, the Z features ought to appear as noise. Even if the corpus-wide distribution of features Z may be quite un-noisy in themselves, there may exist interesting clusterings where they do appear to be noise.

One setting in which this formulation is particularly relevant is those cases in which features bear semantic content. An obvious example would be that of text mining, where individual features

(terms) contain semantic information about the document. An analyst could then select terms associated with the undesired structure. This approach is also useful in more specialized settings, in which it is known a priori that certain features dominate the structure of the data but this structure is not desired.

1.2.1 Contributions

The major contributions from this work are formalization of the non-redundant clustering problem, a suite of techniques designed to solve the problem of non-redundant clustering, and an analysis of their comparative strengths. The suite of techniques consists of:

Constrained Model Based Clustering (CMBC): A model-based approach to non-redundant clustering. We present a novel formulation which enforces non-redundancy by incorporating model-level constraints. From that, an Expectation Maximization algorithm is derived. Obtaining a tractable solution requires assumptions that the data and known structure is binary.

Coordinated Conditional Information Bottleneck (CCIB): An information-theoretic approach which uses the concept of *conditional mutual information* to form an objective which is optimized by finding structure *conditional* on the known known structure. This results in a powerful, flexible framework. We present formulations for various distributional assumptions of the data as well as a version which accepts non-categorical, continuous known structure.

Conditional Ensemble Clustering (*CondEns*): A framework which does not enforce non-redundancy directly via the objective but rather uses consensus-based ensemble clustering methods to efficiently obtain solutions from ensembles of base clustering algorithms conditioned on the known structure. This approach allows for virtually any base clustering technique to be used.

Before detailing these techniques, essential concepts from information theory and clustering will be introduced. This will allow for a formal definition of the problem. Subsequently, the three

techniques will be presented and experimental results discussed.

Chapter 2

Information Theory Background

In this chapter, relevant concepts from information theory are presented. They will be used extensively in characterizing the non-redundant clustering problem and deriving clustering algorithms. Section 2.2 reviews the concept of entropy, Section 2.3 reviews mutual information, and 2.4 discusses an important approximation for entropy and mutual information criteria.

2.1 Information Theory

Information theory provides a method of quantifying information content, for instance, how much information a given clustering contains. A probabilistic setting is assumed and information content is related to the amount of uncertainty of a distribution. An excellent reference on the topic is [Cover and Thomas, 1991], from which the following definitions are drawn.

2.2 Entropy

A crucial concept is that of *entropy*, which is a measure of the uncertainty of a random variable.

Definition 1. *The* Shannon entropy*, $H(X)$, of a discrete random variable X is:*

$$H(X) = -\sum_{x \in \mathscr{X}} p(x) \log_2 p(x). \tag{2.2.1}$$

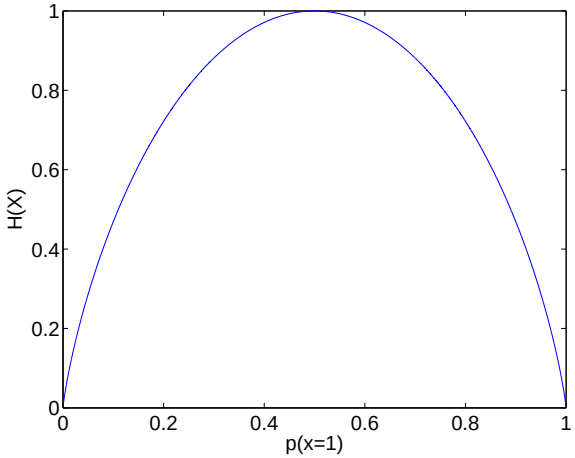


Figure 2.1: The entropy of Bernoulli variable $x \in \{0, 1\}$ as $p(x = 1)$ varies in [0,1].

The entropy of a Bernoulli variable $x \in \{0, 1\}$ is depicted in Figure 2.1. Uncertainty should be greatest in the case of the uniform distribution: $p(x = 0) = p(x = 1) = 1/2$. Not surprisingly for this value, entropy is maximized. As the distribution becomes less balanced, the entropy decreases.

Entropy has several properties which will be of use:

1. Entropy is non-negative: $H(X) \geq 0$.

2. Entropy is bounded: $H(X) \leq \log |\mathscr{X}|$.

3. Entropy is concave: $H''(X) \leq 0$.

As discussed so far, entropy is defined over a single variable X. The definition of entropy can be extended to address collections of variables via the *joint entropy* and *conditional entropy*. The joint entropy measures the uncertainty jointly over a set of variables and the conditional entropy measures the uncertainty of one variable given that another is known.

Definition 2. *The* joint entropy *of variables $X_1, X_2, \ldots, X_n$ is given by:*

$$H(X_1, X_2, \ldots, X_n) = -\sum_{x_1 \in \mathscr{X}_1} \sum_{x_2 \in \mathscr{X}_2} \cdots \sum_{x_n \in \mathscr{X}_n} p(x_1, x_2, \ldots, x_n) \log p(x_1, x_2, \ldots, x_n). \tag{2.2.2}$$

Definition 3. *The* conditional entropy *of variable X conditioned on Y is given by:*

$$H(X|Y) = \sum_{x \in \mathscr{X}} \sum_{y \in \mathscr{Y}} p(x,y) \log p(x|y). \tag{2.2.3}$$

The conditional entropy expresses the uncertainty of variable X given that the value of Y is known.

2.3 Mutual Information and Relative Entropy

Definition 4. *The* mutual information *of variables X and y is given by:*

$$I(X;Y) = \sum_{x \in \mathscr{X}} \sum_{y \in \mathscr{Y}} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}. \tag{2.3.1}$$

Often, mutual information is given in terms of the conditional probability:

$$I(X;Y) = \sum_{x \in \mathscr{X}} \sum_{y \in \mathscr{Y}} p(x,y) \log \frac{p(x|y)}{p(x)} = \sum_{y \in \mathscr{Y}} \sum_{x \in \mathscr{X}} p(x,y) \log \frac{p(y|x)}{p(y)}. \tag{2.3.2}$$

Mutual information may be written in terms of entropies:

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{2.3.3}$$

A number of useful properties follow:

1. Mutual information is symmetric: $I(X;Y) = I(Y;X)$.

2. Mutual information is non-negative: $I(X;Y) \geq 0$.

3. Mutual information is bounded: $I(X;Y) \leq \min(H(X), H(Y))$.

The definition of mutual information may be extended to any number of variables. This was first called "multiinformation" by Perez who uses the quantity in [Perez, 1977], though the quantity also appears in earlier work such as [Watanabi, 1960].

Definition 5. *The* multiinformation *of variables $X_1, X_2, \ldots, X_n$ is given by:*

$$I(X_1; X_2; \ldots; X_n) = \sum_{x_1 \in \mathscr{X}_1} \sum_{x_2 \in \mathscr{X}_2} \cdots \sum_{x_n \in \mathscr{X}_n} p(x_1, x_2, \ldots, x_n) \log \frac{p(x_1, x_2, \ldots, x_n)}{p(x)p(x_2) \cdots p(x_n)}. \tag{2.3.4}$$

There is also the notion of conditional mutual information, which is the mutual information of X and Y conditional on Z. This measures how much incremental knowledge X offers about Y assuming that Z is known.

Definition 6. *The* conditional mutual information *of variables X and Y given Z is:*

$$I(X;Y|Z) = \sum_{x\in\mathscr{X}} \sum_{y\in\mathscr{Y}} \sum_{z\in\mathscr{Z}} p(x,y,z) \log \frac{p(x,y|z)}{p(x|z)p(y|z)}. \tag{2.3.5}$$

Another useful quantity is the *Kullback Leibler (KL) divergence*, also known as the *relative entropy* which measures the divergence between two probability distributions:

Definition 7. *The* Kullback Leibler (KL) divergence *for distributions $p(x)$ and $q(x)$ is:*

$$D_{KL}[p||q] = \sum_{x\in\mathscr{X}} p(x) \log \frac{p(x)}{q(x)}. \tag{2.3.6}$$

The KL-divergence is always non-negative and is equal to 0 if and only if $p = q$. Note that it is not a true metric as it is not symmetric and does not satisfy the triangle equality.

2.4 Generalized Entropy

The *structural* or *generalized entropy of degree s* is an approximation of the Shannon entropy due to [Havrda and Charvát, 1967].

Definition 8. *The* generalized entropy of degree s *where $s > 0, s \neq 1$ is defined as:*

$$H^s(X) = (2^{1-s} - 1)^{-1} \left(\sum_{x\in\mathscr{X}} p(x)^s - 1 \right). \tag{2.4.1}$$

Of particular note is the quadratic version, for $s = 2$, which in certain algorithmic settings may be more convenient than the Shannon entropy:

$$H^2(X) = 2 \left(1 - \sum_{x\in\mathscr{X}} p(x)^2 \right). \tag{2.4.2}$$

The quadratic entropy is pictured in Figure 2.2.

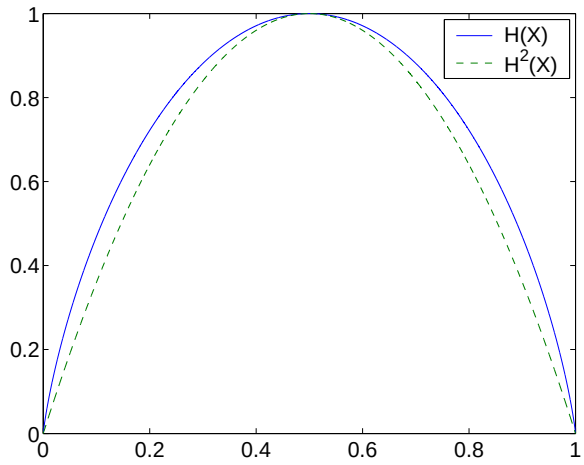


Figure 2.2: Comparison of Shannon Entropy $H(X)$ and Generalized Quadratic Entropy, $H^2(X)$ for $x \in \{0, 1\}$ as $p(x = 1)$ varies in [0,1].

Chapter 3

Clustering Background

3.1 Clustering

We first introduce the notion of a clustering and discuss hard and soft variants. Next, necessary to the definition of a clustering is an explicit criterion by which to measure the quality of a clustering. Such a criterion may in practice give rise to multiple clusterings of interest in a data set. This phenomenon provides the motivation for the problem of non-redundant data mining.

3.2 Clusterings

The most common definition of a clustering is as a *hard clustering*, where each item is assigned to exactly one cluster. Mathematically it is defined as follows:

Definition 9. *Given a dataset X consisting of N items: $X = \{x_1, x_2, \ldots x_N\}$, a* hard clustering C *is a partition of X into K sets $\{c_1, c_2, \ldots c_K\}$.*

As C is a partition of X, the following two properties hold:

i. C is disjoint:

$$c_k \cap c_l = \emptyset \qquad \text{for } k \neq l.$$

ii. The union of C is X:

$$\bigcup_{k=1}^{K} c_k = X.$$

A clustering C may be represented by an $N \times K$ assignment matrix A where

$$A_{ij} = \begin{cases} 1 & \text{if } x_i \in c_j, \\ 0 & \text{otherwise.} \end{cases} \tag{3.2.1}$$

Equivalently, a clustering may be represented by a function: $c : \{1, 2, \ldots, N\} \rightarrow \{1, 2, \ldots, K\}$ so that $c(i)$ returns the cluster number k to which x_i is assigned.

3.2.1 Soft clusterings

A more general definition of a clustering is that of a *soft clustering*. In a soft clustering, cluster assignments are assumed to be "soft", or probabilistic. Thus, each instance x_i belongs to a cluster c_k with some probability $p(c_k|x_i)$ where $\sum_{k=1}^{K} p(c_k|x_i) = 1$. This definition allows an instance to be assigned, in probability, to multiple clusters.

Definition 10. *Given a dataset X consisting of N items: $X = \{x_1, x_2, \ldots x_N\}$, a* soft clustering *$C$ is a probabilistic assignment, $P(C|X)$ of items X to the K clusters $\{c_1, c_2, \ldots c_K\}$.*

A soft clustering C is typically represented by its distribution $P(C|X)$. A hard clustering is a special case of soft clusterings where, for a given item, all of the probability mass is placed on one c_k. Soft clusterings may be used in problem settings where hard clusterings are not required. Soft clusterings may also be used as an intermediate solution within algorithms which ultimately produce hard clusterings.

3.3 Clustering Criteria

As defined, the number of clusterings for a data set X is the number of partitionings of N objects, which is given by the Stirling number of the second kind, and is computed according to:

$$S(N,K) = \frac{1}{K!}\sum_{i=0}^{K-1}(-1)^i\binom{K}{i}(K-i)^N. \tag{3.3.1}$$

Even for small N and K, the enumeration of partitions becomes intractable. Fortunately, in clustering problems, one is typically interested in finding only those partitions in which similar items are grouped together. Informally, one desires a grouping which maximizes intra-group similarity and minimizes inter-group similarity. This introduces a notion of similarity which must be made explicit. Characterizing similarity may be achieved through the specification of a similarity or distance function between items [Jain and Dubes, 1988] or the definition of an objective function which is to be optimized [MacQueen, 1967]. The choice of similarity or objective function must be tailored to the representation and modeling of the data items but may also be designed in order to favor certain types of clusterings over others. For instance, one may choose a function which favors spherical groups of data items[MacQueen, 1967] or linear groups of data items[Zhang, 2003]. We assume a general objective function $F(C,X)$ which is to be maximized. The general clustering problem can then be given as:

Definition 11. *The* clustering problem *is given for data X and objective function F as:*

$$\max_C F(C,X) \tag{3.3.2}$$

$$\textit{s.t. } C \textit{ is a valid clustering.} \tag{3.3.3}$$

3.4 Multiple Clusterings and Cluster Orthogonality

It is often the case that there will exist multiple clusterings which obtain high values in the objective function. These may consist of minor variations on a single clustering or may include clusterings

which are substantially dissimilar. Characterizing either case requires some notion of clustering similarity. Multiple techniques exist for determining the similarity or dissimilarity of clusterings. Many of the approaches were originally developed for use as external criteria for the *cluster validation* task, which seeks to measure the effectiveness of a clustering algorithm by comparing the results against a known clustering. Overviews of these methods are provided in [Jain and Dubes, 1988, Meilă, 2003] of which we review two commonly used groups:

Pair-counting criteria These approaches make use of the counts of pairs of points on which two clusterings agree or disagree. Several notable examples are the Rand index [Rand, 1971], the Jaccard index [Jaccard, 1912], and the Hubert index [Hubert and Schultz, 1976]. As discussed in [Meilă, 2003], these criteria suffer from problems in that the base-line score is dependent on the number of points in each cluster and even then, the base-line score requires unrealistic assumptions.

Information theoretic criteria An information-theoretic approach addresses the problem by considering how much information one clustering provides about another. That is, given that a point is in one cluster in clustering C, how much information does that provide about which clustering the point is assigned in clustering Z. The use of mutual information was proposed by [Dom, 2002]. Later [Meilă, 2003] suggested a closely related measure, the *variation of information (VI)*, which is shown to be a true metric over the space of clusterings.

The variation of information criterion will be useful for our purposes as it is a true metric and allows the application of results from information theory.

3.4.1 Variation of Information (VI)

We begin by presenting the definition of Variation of Information as proposed in [Meilă, 2003] and then discuss relevant properties of this metric.

Definition 12. *The* variation of information *between two clusterings C and Z, $VI(C, Z)$ is defined as:*

$$VI(C, Z) = H(C) + H(Z) - 2I(C, Z). \tag{3.4.1}$$

The definition in 3.4.1 can be rewritten as:

$$VI(C, Z) = [H(C) - I(C, Z)] + [H(Z) - I(C, Z)] = H(C|Z) + H(Z|C), \tag{3.4.2}$$

which expresses the quantity as the sum of the conditional entropies. It can be shown that VI is a true metric, as it obeys the following three properties:

(i.) Non-negativity $VI(C, Z) \geq 0$ with equality if and only if $C = Z$.

(ii.) Symmetry $VI(C, Z) = VI(Z, C)$.

(iii.) Triangle Inequality For clusterings C, Z, W,

$$VI(C, Z) + VI(Z, W) \geq VI(C, W). \tag{3.4.3}$$

Further, VI is *bounded*, that is,

Boundedness

$$VI(C, Z) \leq H(C) + H(Z). \tag{3.4.4}$$

Boundedness follows by non-negativity of mutual information and is a consequence of the fact that the space of all clusterings is bounded.

3.5 Clustering Techniques

We now present background on three widely used traditional clustering techniques which will be used in the development of non-redundant clustering algorithms.

3.5.1 K-means

The technique referred to as *k-means clustering* is a simple and popular method of clustering proposed by MacQueen [MacQueen, 1967], also known as *nearest centroid sorting* [Forgy, 1965] or *iterative relocation* [Jancey, 1966]. It initializes a random assignment of instances to clusters and consists of two phases: calculating the means of each cluster and reassigning each instance based on the cluster mean with which it is most similar. Similarity is typically measured via the squared Euclidean distance between an instance x_i and the cluster centroid μ_k:

$$d(x_i, \mu_k) = ||x_i - \mu_k||^2 \tag{3.5.1}$$

The result minimizes the within-point scatter, expressed by the sum-of-squares criterion:

$$J = \sum_{j=1}^{K} \sum_{i:c(i)=k} ||x_i - \mu_j||^2 \tag{3.5.2}$$

As originally proposed, k-means consists of two alternating phases: a centroid-calculation phase and an instance-reassignment phase. In order to distinguish from other update schemes, we refer this approach as *batch k-means*, given in Algorithm 3.1.

Algorithm 3.1 Batch k-means clustering

Randomly initialize cluster assignment $c(i)$
repeat
  // Centroid Computation Phase:
  for $k = 1 \dots K$ **do**
    calculate mean μ_k for cluster k:

$$\mu_k = \frac{1}{\mathsf{size}(k)} \sum_{i:c(i)=k} x_i \tag{3.5.3}$$

  end for
  // Instance Reassignment Phase:
  for $i = 1 \dots N$ **do**
    reassign i:

$$c(i) = \arg\min_k d(x_i, \mu_k) \tag{3.5.4}$$

  end for
until convergence

There is also a variant known as *online k-means*, also referred to as *sequential k-means* in which

data points are reassigned individually, followed by a recalculation of the affected centroids. Pseudocode for sequential k-means is shown in Algorithm 3.2.

Algorithm 3.2 Sequential k-means clustering

```
Randomly initialize cluster assignments c(i)
repeat
  for i = 1...N do
    c_old ← c(i), n_old ← size(c_old)
    // select cluster with minimum d(x_i, c_k):
      c_new ← arg min_k d(x_i, μ_k), n_new = size(c_new)
    // reassign instance and recompute affected means:
      c(i) = c_new
      μ_old = n_old/(n_old − 1) · μ_old − x_i/(n_old − 1)
      μ_new = n_new/(n_new + 1) · μ_new − x_i/(n_new + 1)
  end for
until convergence
```

3.5.2 Expectation Maximization

K-means clustering with square error criterion turns out to be a special case of the *Expectation Maximization (EM)* algorithm of [Dempster et al., 1977]. The EM algorithm is a general algorithm for maximum a posteriori estimation on problems which have incomplete data (which means the values of certain variables are not observed.) We present the algorithm here as applied to the clustering problem where the incomplete data correspond to the unknown class labels of instances.

An instance x_i is assumed to be generated according to a mixture model over the components (classes) c. Each class-conditional distribution is parameterized by θ:

$$P(x_i|\theta) = \sum_{k=1}^{K} P(c_k|\theta)P(x_i|c_k;\theta), \tag{3.5.5}$$

which, combined with the assumption that instances are independently generated, yields the following probability for the entire data set $\mathcal{X}$:

$$P(\mathcal{X}|\theta) = \prod_{i=1}^{N} P(x_i|\theta) = \prod_{i=1}^{N}\sum_{k=1}^{K} P(c_k|\theta)P(x_i|c_k;\theta). \tag{3.5.6}$$

For the moment, we assume the prior $P(\theta)$ is uniform, in which case the maximum a posteriori estimation problem is a maximum likelihood estimation problem. Instead of maximizing $P(\mathcal{X}|\theta)$

directly, we will instead consider maximizing the log-likelihood:

$$\max_{\theta} \log P(\mathcal{X}|\theta) = \sum_{i=1}^{N} \log \sum_{k=1}^{K} P(c_k|\theta) P(x_i|c_k;\theta) \tag{3.5.7}$$

Note that the resulting formulation is a log of sums for which calculating closed-form equations, as used in maximization, is not straightforward. To address this difficulty, a variational method is used. In particular, [Neal and Hinton, 1998] suggest that EM may be viewed as an alternating maximization of

$$\max_{\theta,Q} F(Q,\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} Q(c_k|x_i;\theta) \log P(c_k|\theta) P(x_i|c_k;\theta) - H(Q), \tag{3.5.8}$$

where the alternating steps take turns between optimizing the objective with respect to θ and Q. An advantage of this formulation is that the sum of logs has disappeared so simple closed-form solutions may be derived for a wide variety of distributions. The solutions obtained by differentiating with respect to Q and θ result in the expectation and maximization steps, respectively. These two steps are then combined to produce the EM algorithm, described in Algorithm 3.3.

Algorithm 3.3 Expectation Maximization (EM) algorithm

Randomly initialize parameters $\hat{\theta}$.
repeat
(**E-step**) Use the current parameter $\hat{\theta}$ to estimate membership probabilities:

$$Q(c_k|x_i;\hat{\theta}) \propto P(c_k|\hat{\theta}) P(x_i|c_k;\hat{\theta}) \tag{3.5.9}$$

(**M-step**) Use the $Q(c_k|x_i;\hat{\theta})$ for the maximization of $\hat{\theta}$:

$$\hat{\theta} = \arg\max_{\theta} \log \tilde{P}(\mathcal{X}|\theta) P(\theta) \tag{3.5.10}$$

where

$$\tilde{P}(\mathcal{X}|\theta) = \sum_{i=1}^{N} \sum_{k=1}^{K} Q(c_k|x_i;\hat{\theta}) P(c_k|\theta) \log P(x_i|c_k;\theta) \tag{3.5.11}$$

until convergence

The algorithm produces the class-conditional density $Q(c_k|x_i;\hat{\theta})$ which may then be used for cluster assignment. As given, Q will typically yield soft-assignments. If hard assignments, where each instance is assigned to a single cluster, are desired the EM algorithm may be modified to produce

hard assignments in the E-step. Another approach to obtain hard assignments is to use deterministic annealing. Deterministic annealing, which will be discussed later, is a procedure that begins with uniform soft assignments, and as a temperature parameter is annealed, produces gradually harder assignments.

We note that, as stated earlier, the EM procedure is equivalent to k-means clustering under the assumptions that clusters are modeled by spherical Gaussian distributions, hard assignments to a single cluster are used within the E-step, and the $P(c_k)$ are uniform.

3.5.3 Informational Bottleneck

The *Information Bottleneck (IB)* is an information-theoretic approach which may be applied to tasks such as clustering. Presented in [Tishby et al., 1999], the information bottleneck is a general formulation from which a variety of algorithmic techniques may be derived. It is motivated by rate distortion theory which is concerned with quantization subject to some distortion measure. In contrast with the models discussed previously, information bottleneck assumes three variables: X are variables representing a particular instance of the data set $\mathcal{X}$, Y refers to features, and C is the cluster variables. Using this representation, the instance-conditional distribution $p(y|x)$ can be determined from the data. Note that the distribution $p(y|x)$ is assumed to be estimated from the data and is not a parameter in this case. Instead, the information bottleneck is parameterized by the membership probabilities $p(c|x)$. The goal is then to minimize the functional:

$$\min_{p(c|x)} \mathcal{L} = I(C;X) - \beta I(Y;C) \tag{3.5.12}$$

$$= \sum_{c\in C, x\in x} p(c,x) \log \frac{p(c|x)}{p(c)} - \beta \sum_{y\in Y c\in C} p(y,c) \log \frac{p(y|c)}{p(y)} \tag{3.5.13}$$

The first term in the objective, $I(C;X)$ measures the compactness of the representation; a uniform (soft) assignment to clusters results in mutual information of 0 whereas a hard assignment would maximize $I(C;X)$. In practice, hard assignments are obtained by either taking the objective as β approaches infinity or β is annealed until a hard assignment is obtained. This leaves the second

term, $I(Y;C)$. We observe that the $p(y)$ in the denominator is constant with respect to the data set and may be ignored. This leaves the expression $\sum p(y,c)\log p(y|c)$ which is the equivalent of the log-likelihood term in the maximum likelihood formulation. In fact, [Slonim and Weiss, 2002] show that under certain conditions, the information bottleneck formulation is mathematically equivalent to the maximum likelihood formulation.

The formal solution to the information bottleneck problem may be obtained by treating $p(c|x)$ as a variational parameter and results in the self-consistency equations (3.5.14) shown in Algorithm 3.4, all of which must be satisfied at a minimum of the objective. We assume the algorithm begins with an initial value for the membership distribution $p(c|x)$ that may either be obtained by random initialization, or in the case of a continuation method like deterministic annealing, is obtained from the previous optimization. The algorithm itself consists of alternating between computing the membership distribution and the auxiliary distributions (for which we explicitly use $q()$). The resulting alternating minimization procedure has been shown to converge [Tishby et al., 1999].

The KL divergence which appears in the membership equation (3.5.14c) emerges as the effective distortion measure. If a given $p(y|x)$ is similar to some $q(y|c)$ as measured by the KL divergence, that cluster c should be a good representative of instance x. Correspondingly, (3.5.14c) assigns high probability to instance x belonging to that cluster c: $p(c|x)$. On the other hand, as the KL divergence increases, the membership probability $p(c|x)$ decreases.

Algorithm 3.4 iterative Information Bottleneck (iIB)

Require: β, initial values for parameters $p(c|x)$

repeat

Update auxiliary distributions:

$$q(c) = \textstyle\sum_x p(x)p(c|x) \tag{3.5.14a}$$

$$q(y|c) = \textstyle\sum_x p(y|x)\frac{p(c|x)p(x)}{q(c)} \tag{3.5.14b}$$

Update membership distribution

$$p(c|x) \propto q(c)e^{-\beta D_{KL}(p(y|x)||q(y|c))} \tag{3.5.14c}$$

until convergence

As mentioned earlier, the value of β crucially determines how peaked the $p(c|x)$ will be. Low values of β result in a soft assignment whereas high values of β will assign all the probability mass for an instance to the cluster c with minimum KL divergence. In practice, hard assignments may be desired. We discuss two approaches to obtaining hard assignments and examples of algorithms corresponding to each approach.

Deterministic Annealing Information Bottleneck (daIB)

The parameter β may be used as an annealing parameter resulting in a continuation method which have proven useful in clustering tasks [Rose, 1998]. In our case, β corresponds to the inverse annealing temperature and so a deterministic annealing approach would begin with $\beta = 0$ and increase β at each phase. At $\beta = 0$, all instances are assigned uniformly to all clusters, all of which would be represented by the same feature distribution. This can be seen as effectively representing all the data by a single cluster. As β increases, phase transitions will occur, where an effective cluster bifurcates into separate clusters. In [Slonim, 2002] an elegant deterministic annealing procedure is given which can be used to identify precisely at what values of β these phase transitions occur. We consider a simple deterministic annealing procedure which increases β and for each value of β, executes Algorithm 3.4 to obtain a solution as shown in 3.5.

Algorithm 3.5 Deterministic Annealing Information Bottleneck (daIB) clustering

Require: annealing temperature increase factor $b > 1$
  initialize inverse temperature $\beta = \frac{1}{\rho} = \epsilon$
  initialize $p(c|x)$ randomly
  repeat
    iterate equations from Algorithm 3.4 until convergence
    increase inverse temperature: $\beta = b \cdot \beta$
  until no change in $p(c|x)$ and $\forall c, x : p(c|x) \in \{0, 1\}$ or max iterations reached

Sequential Information Bottleneck (sIB)

It is also possible to obtain hard-clustering algorithms by taking the self-consistency equations (3.5.14) as $\beta \to \infty$. The resulting equations may then be used to construct an agglomerative

clustering method as in [Slonim and Tishby, 2000] or, as we discuss here, a sequential clustering method as proposed in [Slonim et al., 2002]. The sequential approach is similar to that of sequential k-means in Algorithm 3.2, only instead using the information bottleneck objective.

Algorithm 3.6 Sequential Information Bottleneck (sIB) clustering

```
Randomly initialize cluster assignments c(i)
repeat
  for i = 1...N do
    c_old ← c(i)
    // select cluster with minimum d(x_i, c_k):
      c_new ← arg min_k ΔL(c_k, c_old)
    // reassign instance and recompute affected distributions:
      c(i) = c_new
      update q(c), q(y|c) for c ∈ {c_old, c_new}
  end for
until convergence
```

3.6 Ensemble Clustering

Ensemble clustering deals with the problem of combining multiple clusterings to produce a combined clustering solution. This problem has recently received substantial attention (cf. [Topchy et al., 2003, Strehl and Ghosh, 2002]) as a practical way of combining results from different clustering algorithms as well as to avoid overfitting by data resampling [Minaei-Bidgoli et al., 2004]. The goal is, given l clusterings $C = \{C^1, \ldots, C^l\}$ where each clustering C^j partitions the data into k_j clusters, to find a combined partition C into a pre-specified number of k clusters.

In [Topchy et al., 2003], several techniques for obtaining a combined clustering solution are compared. The *median partition* technique, in particular, has the most attractive time complexity while achieving results on par with the other techniques studied. It can be motivated by the objective presented in [Strehl and Ghosh, 2002]:

$$C^* = \arg\max_C \sum_{j=1}^{l} I(C; C^j)\,. \tag{3.6.1}$$

Instead of optimizing the objective directly, [Topchy et al., 2003] solve for a related quantity, the

generalized mutual information. Generalized mutual information follows from the definition of generalized entropy for degree s:

$$H^s(C) = (2^{1-s} - 1)^{-1} \left(\sum_{j=1}^{l} p(c_i)^s - 1 \right), \tag{3.6.2}$$

where $s > 0$ and $s \neq 1$. The generalized mutual information would then be:

$$I^s(C; C') = H^s(C') - H^s(C'|C)\,. \tag{3.6.3}$$

One may consider the quadratic mutual information, where $s = 2$. This criterion is up to a factor of 2 equivalent to the *category utility function* U presented in [Gluck and Corter, 1985]. Moreover, it was shown in [Mirkin, 2001] that maximization of Eq. (3.6.3) for a fixed number of target clusters is equivalent to the minimization of a squared-error criterion. The latter can be solved using standard approximation techniques such as k-means clustering.

Chapter 4

Non-Redundant Clustering

We now make use of the concepts in previous sections to formally define non-redundant clustering. We begin by defining the notion of "information-orthogonality," discuss the two possible forms of known structure and then proceed to define the non-redundant clustering problem. We finish by placing the problem in a historical context.

4.1 Orthogonal Clusterings

The goal of obtaining non-redundant clusterings requires that, intuitively, the clusterings obtained be "orthogonal" in some sense. Here we more formally define the notion of orthogonality for clusterings. In particular, we present the idea of *information-orthogonality* for clusters. Note that this is intended to capture the orthogonality of clusterings and is not to be confused with the parameter information-orthogonality of [Cox and Reid, 1987]. In particular, we define:

Definition 13. *Clusterings C and Z are information-orthogonal with regard to X if:*

$$I(C, Z; X) = I(C; X) + I(Z; X) \tag{4.1.1}$$

A consequence of this is given in the following lemma with proof given in Appendix A.1.

Lemma 1. *If C and Z are information-orthogonal w.r.t. X, then*

$$H(C, Z) = H(C) + H(Z). \tag{4.1.2}$$

As discussed in [Cover and Thomas, 1991], Lemma 1 is a necessary and sufficient condition for the variables C and Z to be independent. This is another way of stating the property (proof in Appendix A.)

Lemma 2. *C and Z are information-orthogonal if and only if C and Z are independent:*

$$P(C, Z) = P(C)P(Z). \tag{4.1.3}$$

A necessary and sufficient condition for independence which we will find useful in practice is:

$$P(C|z_i) = P(C|z_j) \forall C,\ \forall z_i, z_j. \tag{4.1.4}$$

We can further use Lemma 1 in order to relate information-orthogonality with the variation of information (VI) perspective:

Lemma 3. *If C and Z are information-orthogonal w.r.t. X, then $VI(C, Z)$ is maximal:*

$$VI(C, Z) = H(C) + H(Z). \tag{4.1.5}$$

In this section we have presented four equivalent conditions for information-orthogonality which are reviewed in the following definition:

Definition 14. *Clusterings C and Z are* information-orthogonal *with regard to X if and only if:*

(i.) Original definition:

$$I(C, Z; X) = I(C; X) + I(Z; X). \tag{4.1.6}$$

(ii.) Entropy formulation:

$$H(C, Z) = H(C) + H(Z). \tag{4.1.7}$$

(iii.) C and Z are independent:

$$P(C, Z) = P(C)P(Z). \tag{4.1.8}$$

(iv.) The Variation of Information metric, $VI(C, Z)$, is maximal:

$$VI(C, Z) = H(C) + H(Z). \tag{4.1.9}$$

4.2 Non-Redundant Clustering

Categorical and Continuous known structure Z

We allow for two forms of known structure Z. The values for Z may be *categorical*, in which case it provides a categorization of data X. This categorization may be obtained directly from the features of X, a known background classification, or a previously obtained clustering. In other situations, the Z may be *continuous*, meaning the values may be real-valued vectors associated with each instance in X.

We note here that the definitions of information-orthogonality given in Definition 14, while intended to apply to clusterings also apply to continuously valued Z. This allows us to use the information-orthogonal property regardless of whether the Z is categorical or continuous.

Problem Definition

Non-redundant clustering is a clustering problem as in Definition 11 however with the added constraint that the clustering C and Z be information-orthogonal.

Definition 15. *The **non-redundant clustering problem** is given for data X, known structure*

Z, and objective function F as:

$$\max_C \quad F(C, X) \tag{4.2.1}$$

$$s.t. \quad C \text{ is a valid clustering,} \tag{4.2.2}$$

$$C \text{ and } Z \text{ are information-orthogonal.} \tag{4.2.3}$$

This definition is simply the clustering problem from Definition 11 with the addition of an information-orthogonality constraint. One way to intuitively explain the information-orthogonality constraint is using Definition 14(iii.), which implies

$$p(c_i|z_j) = p(c_i|z_k) \qquad \forall i, j, k, \tag{4.2.4}$$

which means that knowing the value of Z does not affect the probability of the cluster assignments. Thus, in the categorical case, possessing the known cluster assignment for an instance does not affect the solution cluster to which it is assigned. In that sense, the two clusterings are independent.

4.3 Motivation and historical context

A commonly identified liability of clustering is the large degree to which the solution returned is dictated by assumptions inherent in the choice of similarity or objective function [Hertz et al., 2004]. It has long been recognized that for exploratory data analysis, a key issue is to select the proper data representation and clustering methodology and objective. In [Jain and Dubes, 1988], it is argued that an iterated cyclical process is necessary in order to obtain useful analysis. The process is identified as a cycle consisting of the following seven stages: data collection, initial screening, representation, clustering tendency, clustering strategy, validation, and interpretation. A user may provide guidance at any of the stages. This guidance may take the form of: controlling the amount of data collected, normalizing the data, selecting an appropriate feature space, assessing the predisposition to cluster of the represented data, selecting a clustering algorithm and similarity or distance function, deciding on a measure for the merit of the obtained clustering, and finally, selecting how

to integrate and present the results of cluster analysis. In our case, we have assumed an adequate representation of the data and so most relevant to our task is the selection of clustering algorithm and, in particular, the objective function which is maximized.

The selection of objective function for clustering may be done by hand using domain knowledge. In [Jain and Dubes, 1988], an interactive approach is suggested by which a series of objective functions are applied and evaluated. Another approach is to use any of a variety of *constrained clustering* approaches which have been recently proposed. They share the characteristic that the user provides some a priori knowledge of a desired solution. One direction is based on the notion of pairwise *instance-level constraints* such as *must-link*, which requires that two instances appear in the same cluster, or *cannot-link*, which requires that two instances appear in different clusters [Wagstaff and Cardie, 2000]. These constraints may be enforced directly as in [Wagstaff and Cardie, 2000, Klein et al., 2002]. Another direction is to learn meaningful distance metrics to use for the clustering based on side information about item similarity [Xing et al., 2002] or relative similarity [Schultz and Joachims, 2003]. These two approaches may be combined as in [Bilenko et al., 2004]. In all of these techniques, the user must have some notion of explicit characteristics of a desired solution.

The particular scenario of non-redundant clustering, in which a user instead is allowed to specify characteristics of what is *not desired* was proposed in [Vaithyanathan and Gondek, 2002a] and [Vaithyanathan and Gondek, 2002b]. The first effective technique which may be used to solve non-redundant clustering problems is the Information Bottleneck with Side Information due to [Chechik and Tishby, 2002]. They argue that data may often contain multiple structures and present an elegant information bottleneck formulation to address this problem. In particular, the model differentiates between relevant and irrelevant variables. The resulting objective function penalizes solutions which are similar to the irrelevant variables. While this approach has a number of applications, it may be applied directly to non-redundant clustering. The only issue is how, in the specific case of non-redundant clustering, to properly weigh the penalty term. The Conditional Information

Bottleneck, which we present in Chapter 6, is an attempt to address this.

Non-redundant clustering then follows in the spirit of the exploratory cyclical process discussed in [Jain and Dubes, 1988]. It differs from the constrained clustering approaches in that it does not assume a priori knowledge of a desired solution. This is particularly important in the setting of exploratory data mining where the emphasis is on the discovery of novel structure. In such settings there is not a priori knowledge of desired solutions from which to draw upon.

Non-redundant clustering also differs from both [Jain and Dubes, 1988] and the constrained clustering approaches in a key regard. In those approaches, there is the implicit assumption that for any useful solution, there exists an objective function which when maximized will obtain that solution. Our approach does not make that assumption. It does not require that the useful solution obtain the maximum value of an objective function. Instead, the useful solution may be secondary structure which is outscored by other dominant solutions. Such an approach acknowledges that it may be impossible to formulate an objective function which corresponds perfectly to the usefulness of a solution.

Chapter 5

Constrained Model-Based Clustering

We first consider a model-based approach to the task of non-redundant clustering. We introduce a model for the generation of features Y and known structure Z. The non-redundancy constraint is introduced as a model-level constraint in a constrained maximum likelihood clustering objective. From this formulation, it is possible to derive a family of Expectation Maximization algorithms.

5.1 Model

This section presents the model used for CMBC. We begin with a review of terms, define the model, and then present a constrained objective formulation. We postpone discussing the computation of distributions used in the model until after presenting the approximation of the constrained objective. The objective formulation will motivate a number of decisions used in the computations.

Definition of Terms

We assume an N-item, dataset $\mathcal{D}$ where the ith item is represented by a vector of features $\mathbf{y}_i$ and its associated known structure $\mathbf{z}_i$. The vectors $\mathbf{y}_i$ and $\mathbf{z}_i$ are taken from the sets of all possible vectors $\mathscr{Y}$ and $\mathscr{Z}$: $\mathbf{y}_i \in \mathscr{Y}$, $\mathbf{z}_i \in \mathscr{Z}$. The vector $d_i \in \mathscr{D}$ is used to refer to the concatenation of relevant features and known structure, $\mathbf{d}_i = [\mathbf{y}_i, \mathbf{z}_i]$. For the items appearing in $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_N\}$, we use the notation $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{z}_N\}$ to refer to the features and $\mathcal{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N\}$ to refer to the known structure. All vectors d_i are assumed to be *binary*, i.e., $D \in \{0,1\}^{M+L}$, an assumption which will later see is necessary for the Expectation Maximization algorithms to remain tractable.

Model Specification

Our interest is in grouping these instances to maximize their similarity while simultaneously searching for a structure that is not already present in the known structure. Intuitively, this suggests a model where $\mathbf{z}_i$ and the clusterings are independent given $\mathbf{y}_i$. Consequently, $\mathbf{y}_i$ are assumed to be drawn from the clusters, while the $\mathbf{z}_i$ are drawn conditioned on $\mathbf{y}_i$. The generative model for this is pictured in Figure 5.1. The probability of an item $\mathbf{d}_i$ is then:

$$p(\mathbf{d}_i) = \sum_{k=1}^{K} p(\mathbf{z}_i|\mathbf{y}_i)p(\mathbf{y}_i|c_k)p(c_k) \tag{5.1.1}$$

So, assuming $\mathcal{D}$ is independently drawn, the probability of observing data $\mathcal{D}$ is:

$$p(\mathcal{D}) = \prod_{i=1}^{N} p(\mathbf{d}_i) = \prod_{i=1}^{N} \sum_{k=1}^{K} p(\mathbf{z}_i|\mathbf{y}_i)p(\mathbf{y}_i|c_k)p(c_k) \tag{5.1.2}$$

and the log-likelihood would be:

$$l(\mathcal{D}) = \sum_{i=1}^{N} \log \sum_{k=1}^{K} p(\mathbf{z}_i|\mathbf{y}_i)p(\mathbf{y}_i|c_k)p(c_k) \tag{5.1.3}$$

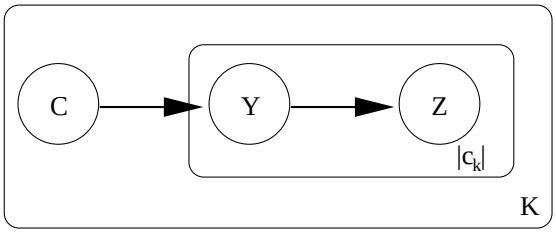


Figure 5.1: Graphical model

5.1.1 Objective Function for CMBC

Here we extend (5.1.3) to incorporate the known structure as model-level constraints. The term *model-level constraints* refers to restrictions on the space of clustering. We begin with the non-redundancy constraint.

Representing Knowledge as Constraints

Assuming the cluster probabilities are equal, a natural way to express the constrained problem, given in Definition 15, is:

$$\max \quad l(\mathcal{D}) \tag{5.1.4}$$

$$s.t. \quad p(c_k|\mathbf{z}_j) = p(c_l|\mathbf{z}_j) \quad \forall j,k,l \qquad \text{where } \mathbf{z}_i, \mathbf{z}_j \text{ is known structure.}$$

Unfortunately, the constraints in (5.1.4) may be too strict and allow only degenerate solutions. To soften the constraints, we replace them with the requirement that the conditional entropy, $H(C|Z)$, be high for

$$H(C|Z) = \sum_{k,i} p(c_k, \mathbf{z}_i) \log p(c_k|\mathbf{z}_i). \tag{5.1.5}$$

Note that $H(C|Z)$ is maximized when the conditional distribution is uniform, which is exactly the constraint in (5.1.4). This results in the following objective function:

$$\mathcal{L} = (1-\gamma) l(\mathcal{D}) + \gamma H(C|Z). \tag{5.1.6}$$

where γ acts as a tradeoff between the loglikelihood and the penalty terms. Intuitively, as γ is increased the resulting solution is farther away from the clusterings associated with Z.

5.1.2 Approximating the Objective Function

Descriptions of how to compute $p(\mathbf{y}_i|c_k)$ and $p(\mathbf{z}_i|c_k)$ will be given later. The entropy term of (5.1.6) may be computed using these values by observing that:

$$\mathcal{L} = (1-\gamma) l(\mathcal{D}) + \gamma H(C|Z) \tag{5.1.7}$$

$$= (1-\gamma) l(\mathcal{D}) + \gamma H(C,Z) - H(Z), \tag{5.1.8}$$

where the $H(C,Z)$ and $H(Z)$ are defined in terms of $p(\mathbf{z}_i|c_k)$, and $p(c_k)$ according to:

$$H(C,Z) = \sum_{k=1}^{K} \sum_{\mathbf{z}_i \in \mathscr{Z}} p(c_k) p(\mathbf{z}_i|c_k) \log p(c_k) p(\mathbf{z}_i|c_k), \tag{5.1.9}$$

$$H(Z) = \sum_{\mathbf{z}_i \in \mathscr{Z}} \left(\sum_{k=1}^{K} p(c_k) p(\mathbf{z}_i|c_k) \right) \log \left(\sum_{r=1}^{K} p(c_r) p(\mathbf{z}_i|c_r) \right). \tag{5.1.10}$$

The objective function in (5.1.8) can be optimized using gradient ascent or Newton-Raphson techniques, however our experiments found this to be slow in practice. Instead, we now describe approximations which allow an Expectation Maximization algorithm to be derived.

Bounding Log-of-Sum Terms

The remaining hurdle in developing an EM algorithm is that each term contains a log of sums, which prevents us from deriving a closed-form solution in order to maximize (5.1.8): the $H(Z)$ sums over k within the log term. As will be seen in Section 5.1.3, the $p(\mathbf{z}_i|c_k)$, which occurs in the log terms of both $H(Z)$ and $H(C,Z)$, also requires performing a sum. We address this issue by replacing these entropy terms with quadratic bounds. Then, closed-form solutions may be derived.

The Shannon entropies in (5.1.9) and (5.1.10) can be approximated by the commonly-used Havrda-Charvát structural α-entropy[Havrda and Charvát, 1967] to obtain quadratic expressions, however this approximation can be quite loose[1]. Instead we make use of quadratic bounds recently presented in [Harremoës and Topsøe, 2001] and [Topsøe, 2003]. These lower and upper bounds are

[1] We attempted to use this approximation in our experiments but found the approximations to differ substantially from the Shannon entropies and result in very poor performance.

built around the *index of coincidence (IC)*, the probability of drawing the same element in two independent trials. From [Topsøe, 2003], we obtain the lower bound which we denote by $\mathrm{H}^{\mathrm{l}}(X)$:

$$H(X) \geq \mathrm{H}^{\mathrm{l}}(X) \doteq \delta_d - \beta_d IC(X) \tag{5.1.11}$$

$$= \delta_d - \beta_d \sum_x p(x)^2, \tag{5.1.12}$$

where d is the number of elements x, used to define:

$$\delta_d = \ln(d+1) + d\ln(1+\frac{1}{d}), \tag{5.1.13}$$

$$\beta_d = (d+1)d\ln(1+\frac{1}{d}). \tag{5.1.14}$$

From [Harremoës and Topsøe, 2001], we use the upper bound on $H(X)$, which we denote as $\mathrm{H}^{\mathrm{u}}(X)$:

$$H(X) \leq \mathrm{H}^{\mathrm{u}}(X) \doteq (\ln d) \cdot \left(1 - \frac{1}{1-\frac{1}{d}} \left(\sum_x p(x)^2 - \frac{1}{d}\right)\right). \tag{5.1.15}$$

Applying (5.1.12) to $H(Z,C)$ and (5.1.15) to $H(Z)$ in our objective function (5.1.8) we obtain:

$$\mathcal{L} \geq (1-\gamma) l(\mathcal{D}) + \gamma \left(\mathrm{H}^{\mathrm{l}}(Y,C) - \mathrm{H}^{\mathrm{u}}(Y)\right). \tag{5.1.16}$$

Simplifying the Likelihood

We describe how to compute $p(\mathbf{z}_i|c_k)$ and the terms in $l(\mathcal{D})$ in Section 5.1.3. For this paper we assume the $p(c_k)$ are constant. From the computation in (5.1.26), it is clear that $p(\mathbf{z}_i|\mathbf{y}_i)$ is constant for a given data set and so can be ignored in calculating $\mathcal{L}$:

$$\mathcal{L} \geq (1-\gamma) l(\mathcal{Y}) + \gamma \left(\mathrm{H}^{\mathrm{l}}(Y,C) - \mathrm{H}^{\mathrm{u}}(Y)\right). \tag{5.1.17}$$

Proof is given in Appendix B.2.

Restricting Entropy Calculation to Those z Present in the Data

Moreover, from (5.1.26), we see the $p(\mathbf{z}_i|\mathbf{y}_h)$ can be estimated only for those $\mathbf{z}_i$ in the data set $\mathcal{Z}$, which requires that the summation in the entropy terms $\mathrm{H}^{\mathrm{l}}(C,Z)$ and $\mathrm{H}^{\mathrm{u}}(Z)$ be restricted to those

$\mathbf{z}_i \in \mathcal{Z}$. We denote these approximations as $\tilde{\mathrm{H}}^{\mathrm{l}}(C, Z)$ and $\tilde{\mathrm{H}}^{\mathrm{u}}(Z)$ which are defined as:

$$\tilde{\mathrm{H}}^{\mathrm{l}}(C, Z) = \left(\delta_{|\mathcal{Z}|} - \beta_{|\mathcal{Z}|} \sum_{\mathbf{z}_i \in \mathcal{Z}} \sum_{k=1}^{K} \left(p(\mathbf{z}_i|c_k) p(c_k) \right)^2 \right) \tag{5.1.18}$$

$$\tilde{\mathrm{H}}^{\mathrm{u}}(Z) = (\log |\mathcal{Z}|) \cdot \left(1 - Q \sum_{\mathbf{z}_i \in \mathcal{Z}} \left(\sum_{k=1}^{K} p(\mathbf{z}_i|c_k) p(c_k) \right)^2 \right). \tag{5.1.19}$$

where $Q = \frac{1}{1 - \frac{1}{|\mathcal{Z}|}}$ and where $|\mathcal{Z}|$ is the number of unique $\mathbf{z}$ which occur in the data set. The $\mathrm{H}^{\mathrm{l}}(C, Z)$ and $\mathrm{H}^{\mathrm{u}}(Z)$ terms of (5.1.17) are replaced with $\tilde{\mathrm{H}}^{\mathrm{l}}(C, Z)$ and $\tilde{\mathrm{H}}^{\mathrm{u}}(Z)$ to produced the final approximation of the objective function, $\tilde{\mathcal{L}}$:

$$\mathcal{L} \geq (1 - \gamma) l(\mathcal{Y}) + \gamma \left(\tilde{\mathrm{H}}^{\mathrm{l}}(Y, C) - \tilde{\mathrm{H}}^{\mathrm{u}}(Y) \right). \tag{5.1.20}$$

Before deriving a solution for the optimization problem, it is first necessary to define precisely how the $p(\mathbf{y}_i|c_k)$ and $p(\mathbf{z}_i|c_k)$ may be calculated.

5.1.3 Computation of $p(\mathbf{y}_i|c_k)$, $p(\mathbf{z}_i|\mathbf{y}_j)$, and $p(\mathbf{z}_i|c_k)$

The parameters for this model are $p(y_{\cdot j}|c_k)$ and $p(c_k)$. The $p(y_{\cdot j}|c_k)$ represent the *class-conditional probabilities*, that is the probability of the jth feature of Y, which we write as $y_{\cdot j}$, conditioned on membership in cluster c_k. The $p(c_k)$, which represent the *class probabilities* are the probability of membership in a given cluster. In developing this algorithm, we will assume that the $p(c_k)$ are uniform over k.

Computation of $p(\mathbf{y}_i|c_k)$

The conditional probabilities $p(\mathbf{y}_i|c_k)$, known as class-conditional probabilities, describe the probability of features given a cluster. We assume a standard Naive Bayes model which asserts the features of Y are generated independently, given cluster membership C:

$$p(\mathbf{y}_i|c_k) = \prod_{j=1}^{M} p(\mathbf{y}_{ij}|c_k) \tag{5.1.21}$$

$$= \prod_{j=1}^{M} p(y_{\cdot j}|c_k)^{\mathbf{y}_{ij}} (1 - p(y_{\cdot j}|c_k))^{1 - \mathbf{y}_{ij}}. \tag{5.1.22}$$

Computation of $p(\mathbf{z}_i|\mathbf{y}_i)$

The conditional probabilities $p(\mathbf{z}_i|\mathbf{y}_i)$ describe the probability of known structure features $\mathbf{z}_i$ occurring given the presence of features $\mathbf{y}_i$. In selecting a $\mathbf{y}_i$, one then probabilistically incurs the presence of certain associated known structure features. In the case of document clustering with categorical known structure, this would imply that the terms in a document probabilistically trigger the known category to which the document belongs. Thus, the $p(\mathbf{z}_i|\mathbf{y}_i)$ play an important role in the constraints of the optimization problem by linking the choice of Y with the associated Z. While one might be tempted to treat the $p(\mathbf{z}_i|\mathbf{y}_i)$ as a free parameter in the optimization, it would allow the possibility of degenerate solutions in which arbitrary values for $p(\mathbf{z}_i|\mathbf{y}_i)$ may be selected in order to satisfy the constraints.

Instead, the $p(\mathbf{z}_i|\mathbf{y}_i)$ are estimated empirically from the data to capture the relationship between Y and Z, and then fixed throughout the optimization process. There are two issues which must be addressed when estimating these quantities. First, the data is typically sparse (i.e. only a small portion of all possible $\mathbf{z}_i \in \mathscr{Z}$ will occur in the dataset $\mathcal{Z}$, and likewise for $\mathscr{Y}$ and $\mathcal{Y}$.) Second, we would like a formulation which accommodates a computationally tractable algorithm.

It is obvious that estimating the $p(\mathbf{z}_i|\mathbf{y}_i)$ based on simple counts is insufficient to address the sparsity issue. The cardinality of arbitrary $\mathcal{Y}$ and $\mathcal{Z}$ may be infinite. Even with the assumption that the data is binary, for even low-dimensional data sets, there would be insufficient data to obtain reasonable estimates of the $2^M \cdot 2^L$ combinations.

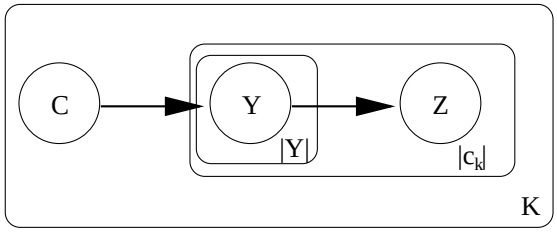


Figure 5.2: Graphical model: Mixture assumption

Instead, we propose a model which assumes that the noise feature vector $\mathbf{z}_i$ is generated according to a mixture of independent features where each feature's presence or absence independently contributes to generation of the known structure Z:

$$p(\mathbf{z}_i|\mathbf{y}_h) = \sum_{j=1}^{M} p(y_{hj})p(\mathbf{z}_i|y_{hj}) \tag{5.1.23}$$

$$= \sum_{j=1}^{M} p(y_{hj})p(\mathbf{z}_h|y_{\cdot j} = 1)^{y_{hj}} p(\mathbf{z}_h|y_{\cdot_j} = 0)^{1-y_{hj}}. \tag{5.1.24}$$

which follows by the assumption that the data is binary and where typically, $p(y_{ij}) = M$, to weigh the features uniformly. This modifies the graphical model in Figure 5.1 to produce the model in Figure 5.2. It now remains to estimate $p(\mathbf{z}_h|y_{\cdot j} = 1)$ and $p(\mathbf{z}_h|y_{\cdot j} = 0)$, which is done by using counts from the data:

$$p(\mathbf{z}_i|y_{\cdot j} = 0) = \frac{|\{\mathbf{d}_t \in \mathcal{D} : \mathbf{z}_t = \mathbf{z}_i \text{ and } \mathbf{y}_{tj} = 0\}|}{|\{\mathbf{d}_t \in \mathcal{D} : \mathbf{y}_{tj} = 0\}|}, \tag{5.1.25}$$

$$p(\mathbf{z}_i|y_{\cdot j} = 1) = \frac{|\{\mathbf{d}_t \in \mathcal{D} : \mathbf{z}_t = \mathbf{z}_i \text{ and } \mathbf{y}_{tj} = 1\}|}{|\{\mathbf{d}_t \in \mathcal{D} : \mathbf{y}_{tj} = 1\}|}. \tag{5.1.26}$$

Computation of $p(\mathbf{z}_i|c_k)$

The $p(\mathbf{z}_i|\mathbf{y}_i)$ are fixed, so the $p(\mathbf{z}_i|c_k)$ are consequences of the choice of $p(y_{\cdot j}|c_k)$ as can be seen in Lemma 4.

Lemma 4. *The $p(\mathbf{z}_i|c_k)$ may be expanded as:*

$$p(\mathbf{z}_i|c_k) = \frac{1}{M} \sum_{j}^{M} \sum_{y_{\cdot j} \in \{0,1\}} p(\mathbf{z}_i|y_{\cdot j})p(y_{\cdot j}|c_k). \tag{5.1.27}$$

The complete proof is given in Appendix B.1. It follows by first marginalizing over all possible $\mathbf{y}_h$ upon which the following is obtained:

$$p(\mathbf{z}_i|c_k) = \sum_{\mathbf{y}_h \in \mathscr{Y}} p(\mathbf{z}_i|\mathbf{y}_h)p(\mathbf{y}_h|c_k). \tag{5.1.28}$$

Then, substituting the definitions from (5.1.22) and (5.1.24), restating the summation as used in (5.1.27), grouping terms, and simplifying the expression produces the final result. Significantly, because of the mixture assumption, a properly normalized distribution is produced which requires

marginalization over the individual *features* $y_{\cdot j}$ instead of marginalization over all possible *feature vectors* $\mathbf{y}_i$. As the dimension of Y, is much smaller than the size of all possible Y vectors $\mathscr{Y}$, the mixture assumption is necessary to ensure tractability.

5.1.4 Final Formulation of the Objective Function

Using the definitions of $p(\mathbf{y}_i|c_k)$, $p(\mathbf{z}_i|\mathbf{y}_j)$, and $p(\mathbf{z}_i|c_k)$, the objective may be stated explicitly.

Expanding the Likelihood Term

The log-likelihood term $l(\mathcal{Y})$ of (5.1.3) is expanded with the familiar:

$$l(\mathcal{Y}) = \sum_{i=1}^{N} \log \sum_{k=1}^{K} p(\mathbf{y}_i|c_k)p(c_k). \tag{5.1.29}$$

As we have assumed that the Y are binary, this results in:

$$l(\mathcal{Y}) = \sum_{i=1}^{N} \log \sum_{k=1}^{K} \frac{1}{M} \sum_{j}^{M} p(\mathbf{z}_i|y_{\cdot j}=1)^{\mathbf{y}_{ij}} p(\mathbf{z}_i|y_{\cdot j}=0)^{(1-\mathbf{y}_{ij})} p(c_k). \tag{5.1.30}$$

Expanding the Entropy Terms

By substituting the definitions from Section 5.1.3 and simplifying, one arrives at:

$$\tilde{\mathrm{H}}(C|Z) \geq \tilde{\mathrm{H}}^{\mathrm{l}}(C,Z) - \tilde{\mathrm{H}}^{\mathrm{u}}(Z) \tag{5.1.31}$$

$$= \left(\delta_{|\mathcal{Z}|} - \beta_{|\mathcal{Z}|} IC(C,Z)\right) \tag{5.1.32}$$

$$= \alpha_{|\mathcal{Z}|} - \beta_{|\mathcal{Z}|} \sum_{k=1}^{K} p(c_k)^2 \frac{1}{M^2} \cdot \left\{ \sum_{j,l} A_2(j,l)\theta_{jk}\theta_{lk} + 2\sum_{j} A_1(j)\theta_{jk} + A_0 \right\} \tag{5.1.33}$$

$$- (\ln|\mathcal{Z}|) \cdot \left(1 - Q \sum_{k=1}^{K}\sum_{r=1}^{K} p(c_k)p(c_r) \frac{1}{M^2} \right. \tag{5.1.34}$$

$$\left. \cdot \left\{ \sum_{j,l} A_2(j,l)\theta_{jk}\theta_{lr} + 2\sum_{j} A_1(j)\theta_{jk} + A_0 \right\} - \frac{Q}{|\mathcal{Z}|} \right). \tag{5.1.35}$$

where the A constants are calculated according to:

$$A_2(j,l) = \sum_{i=1}^{N} \left(p(\mathbf{z}_i|y_{\cdot j}=1) - p(\mathbf{z}_i|y_{\cdot j}=0)\right)\left(p(\mathbf{z}_i|y_{\cdot l}=1) - p(\mathbf{z}_i|y_{\cdot l}=0)\right), \tag{5.1.36}$$

$$A_1(j) = \sum_{i=1}^{N}\sum_{l=1}^{M} p(\mathbf{z}_i|y_{\cdot l}=0)(p(\mathbf{z}_i|y_{\cdot j}=1) - p(\mathbf{z}_i|y_{\cdot j}=0)), \tag{5.1.37}$$

$$A_0 = \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{l=1}^{M} p(\mathbf{z}_i|y_{\cdot j}=0)p(\mathbf{z}_i|y_{\cdot l}=0). \tag{5.1.38}$$

The complete derivation is in Appendix B.3. Note that the parameters in $\tilde{\mathrm{H}}^{\mathrm{l}}(C,Z)$ and $\tilde{\mathrm{H}}^{\mathrm{u}}(Z)$ leave a quadratic polynomial. This will ensure the derivation of simple closed-form M-step equations within the EM algorithm.

Complete Objective Function

Using (5.1.29) and (5.1.35), the complete objective function may be expressed:

$$\begin{aligned}\tilde{\mathcal{L}} =& \sum_{i=1}^{N} \log \sum_{k=1}^{K} \frac{1}{M} \sum_{j}^{M} p(\mathbf{z}_i|y_{\cdot j}=1)^{\mathbf{y}_{ij}} p(\mathbf{z}_i|y_{\cdot j}=0)^{(1-\mathbf{y}_{ij})} p(c_k) \\ &+ \alpha_{|\mathcal{Z}|} - \beta_{|\mathcal{Z}|} \sum_{k=1}^{K} p(c_k)^2 \frac{1}{M^2} \cdot \left\{ \sum_{j,l} A_2(j,l)\theta_{jk}\theta_{lk} + 2\sum_{j} A_1(j)\theta_{jk} + A_0 \right\} \\ &- (\ln|\mathcal{Z}|) \cdot \left(1 - Q \sum_{k=1}^{K}\sum_{r=1}^{K} p(c_k)p(c_r) \frac{1}{M^2} \cdot \left\{ \sum_{j,l} A_2(j,l)\theta_{jk}\theta_{lr} + 2\sum_{j} A_1(j)\theta_{jk} + A_0 \right\} - \frac{Q}{|\mathcal{Z}|} \right).\end{aligned} \tag{5.1.39}$$

5.2 EM Algorithm

We now present a family of EM algorithms which are derived from the model of the preceding section.

Basic EM Algorithm

It is now possible to derive an Expectation Maximization (EM) algorithm to optimize the objective function from (5.1.39). The complete derivation is presented in Appendix B.4. The derivation is similar to that of conventional EM, however in this case there is the added complication of the

entropy term. Recall that EM can be seen as an alternating algorithm, consisting of an expectation step (E-step), which computes the expectation of the hidden variables (in this case, the cluster assignment variables), and a maximization step (M-step), which maximizes the parameters given this expectation. With the objective function (5.1.39), the E-step is unchanged from unconstrained EM, whereas the M-step now requires the solution of a quadratic equation. Both steps are shown in the basic EM algorithm in Algorithm 5.1.

Algorithm 5.1 CMBC basic EM procedure: Partial Maximization (CMBC-pm)

Loop until converged:

For $i = \{1 \dots N\}$:

For $k = \{1 \dots K\}$:

E-step: Compute assignment expectations:

- For all instances $i = 1 \dots N$:

$$q(c_k|\mathbf{d}_i) \propto p(c_k)p(\mathbf{y}_i|c_k) \tag{5.2.1}$$

M-step: Maximize $p(y_{.j}|c_k)$ by solving:

$$F_3 p(\mathbf{y}_h|c_k)^3 + F_2 p(\mathbf{y}_h|c_k)^2 + F_1 p(\mathbf{y}_h|c_k) + F_0 = 0 \tag{5.2.2}$$

.

The F terms will be defined below. As mentioned, the E-step is the familiar expression from unconstrained EM. The M-step finds the optimal value for the model parameters, $p(y_{.j}|c_k)$, given the class assignments. The M-step for the class-conditional probabilities $p(y_{.j}|c_k)$ is obtained according to the following lemma:

Lemma 5. *The maximum likelihood estimate of* $p(y_{.j}|c_k)$ *are obtained by solving the cubic equation:*

$$F_3 p(\mathbf{y}_h|c_k)^3 + F_2 p(\mathbf{y}_h|c_k)^2 + F_1 p(\mathbf{y}_h|c_k) + F_0 = 0 \tag{5.2.3}$$

for coefficients F:

$$F_3 = \frac{-2\gamma p(c_k)^2}{M^2}\left(Q\ln|\mathcal{Z}| - \alpha_{|\mathcal{Z}|}\right)A_2(h,h), \tag{5.2.4}$$

$$\begin{aligned} F_2 =& \frac{2\gamma p(c_k)^2}{M^2}\left(Q\ln|\mathcal{Z}| - \alpha_{|\mathcal{Z}|}\right)A_2(h,h) \\ &+ \frac{2\ln|\mathcal{Z}|\cdot Qp(c_k)}{M^2}\left(\sum_{r,l:(r,l)\neq(k,h)} p(c_r)A_2(h,l)\theta_{lk} + A_1(h)\right) \\ &- \frac{2\ln|\mathcal{Z}|\cdot\beta_{|\mathcal{Z}|}p(c_k)^2}{M^2}\left(\sum_{l:l\neq h} A_2(h,l)\theta_{lk} + A_1(h)\right), \end{aligned} \tag{5.2.5}$$

$$\begin{aligned} F_1 =& -(1-\gamma)\sum_{i=1}^{|\mathcal{D}|} q(c_k|\mathbf{d}_i) \\ &+ \frac{2\ln|\mathcal{Z}|\cdot Qp(c_k)}{M^2}\left(\sum_{r,l:(r,l)\neq(k,h)} p(c_r)A_2(h,l)\theta_{lk} + A_1(h)\right) \\ &- \frac{2\ln|\mathcal{Z}|\cdot\beta_{|\mathcal{Z}|}p(c_k)^2}{M^2}\left(\sum_{l:l\neq h} A_2(h,l)\theta_{lk} + A_1(h)\right), \end{aligned} \tag{5.2.6}$$

$$F_0 = (1-\gamma)\sum_{i=1}^{|\mathcal{D}|} q(c_k|\mathbf{d}_i)y_{ih}. \tag{5.2.7}$$

This result is easily obtained by noting that the function is concave in $p(y_{\cdot j}|c_k)$ and maximizing. The result is a cubic equation which has a closed-form solution. Complete details of the derivation are in Appendix B.4.

Partial Maximization versus Batch Update

The algorithm shown in Algorithm 5.1 performs partial *partial maximization* updates. That is, only one such $p(y_{\cdot j}|c_k)$ is updated in the M-step (5.2.2) before the E-step (5.2.1) is performed. This differs from unconstrained EM, wherein the entire batch of $p(y_{\cdot j}|c_k)$ for all such j and k are updated before performing the E-step. We refer to this as *batch update*. The partial maximization is computationally less attractive, but follows from the derivation because the updated parameters participate in the computation of the F constants. Still, at the expense of potentially suboptimal

solutions, the optimization may be approximated by a batch update which achieves more attractive computational complexity. The batch update algorithm is depicted in Algorithm 5.2.

Algorithm 5.2 CMBC basic EM procedure: Batch Update (CMBC-bu)

Loop until converged:

E-step: Compute assignment expectations:

For $i = \{1 \dots N\}$:

For $k = \{1 \dots K\}$:

$$q(c_k|\mathbf{d}_i) \propto p(c_k)p(\mathbf{y}_i|c_k) \tag{5.2.8}$$

M-step:

For $i = \{1 \dots N\}$:

For $k = \{1 \dots K\}$:

Maximize $p(y_{\cdot j}|c_k)$ by solving:

$$F_3 p(\mathbf{y}_h|c_k)^3 + F_2 p(\mathbf{y}_h|c_k)^2 + F_1 p(\mathbf{y}_h|c_k) + F_0 = 0 \tag{5.2.9}$$

.

Deterministic Annealing

In [Rose, 1998], the use of deterministic annealing for clustering is proposed and derived from basic information-theoretic principles. The chief benefit in considering deterministic annealing approaches in this setting is their tendency toward avoiding local maxima. This behavior produces higher quality solutions. Deterministic annealing approaches are applied in a soft-clustering setting by introducing a temperature parameter T which is initialized to a high temperature. At high values of T, the cluster assignments are smooth (meaning each instance is assigned to all clusters with equal probability.) The T is then reduced according to $T = \alpha T$ for some annealing rate $0 < \alpha < 1$. As T is cooled, the cluster assignments harden as the probability mass is typically centered on one cluster. The deterministic annealing versions of partial maximization and batch update are featured in Algorithms 5.3 and 5.4.

Algorithm 5.3 CMBC Deterministic Annealing EM Algorithm: Partial Maximization (dCMBC-pm)

Select annealing rate $\alpha < 1$. Initialize T to be high. Randomly initialize $p(y_{.j}|c_k)$.

Loop until hard-assignment obtained:

1. Loop until converged:

   For $j = \{1 \dots M\}$

   For $k = \{1 \dots K\}$:

   E-Step Compute assignment expectations:

   - for all instances $i = 1 \dots N$:

$$q(c_k|\mathbf{d}_i) \propto p(c_k)p(\mathbf{y}_i|c_k)^{1/T} \tag{5.2.10}$$

   M-step Maximize for $p(y_{.j}|c_k)$ by solving:

$$F_3 p(\mathbf{y}_h|c_k)^3 + F_2 p(\mathbf{y}_h|c_k)^2 + F_1 p(\mathbf{y}_h|c_k) + F_0 = 0 \tag{5.2.11}$$

2. Decrease temperature $T \leftarrow \alpha T$

5.2.1 Convergence and Complexity

We first note that convergence to a local maximum is guaranteed by the variational solution in Appendix B.4 just as in unconstrained EM. The number of iterations of EM before convergence can vary considerably between initializations and data sets as well as which variant of the EM algorithm is used. Let I be the number of iterations for a given data set and random initialization. Each iteration of the basic partial-maximization algorithm in Algorithm 5.1 contributes $O(KN)$ from the M-step and $O(KN^2)$ from the E-step, for a total runtime of $O(I(KN + KN^2)) = O(IKN^2)$. The basic batch-update algorithm, by contrast, differs in that the E-step contributes only $O(KN)$ for a total runtime of $O(IKN)$. The deterministic annealing variants introduce a number of annealing phases which we denote as P, resulting in runtimes of $O(PIKN^2)$ and $O(PIKN)$ for partial-maximization and batch-update, respectively. As the P and I can vary widely due to the algorithm chosen, we will rely more on experimental analysis in order to judge the tradeoffs among the algorithms.

Algorithm 5.4 CMBC Deterministic Annealing EM Algorithm: Batch Update (dCMBC-bu)

Select annealing rate $\alpha < 1$. Initialize T to be high. Randomly initialize $p(y_{.j}|c_k)$.

Loop until hard-assignment obtained:

1. Loop until converged:

   E-Step Compute assignment expectations:

   - for $i = 1 \dots N$:
     - for $k = \{1 \dots K\}$

$$q(c_k|\mathbf{d}_i) \propto p(c_k)p(\mathbf{y}_i|c_k)^{1/T} \tag{5.2.12}$$

   M-step For $k = \{1 \dots K\}$

   - $forj = \{1 \dots M\}$:
     - Maximize for $p(y_{.j}|c_k)$ by solving:

$$F_3 p(\mathbf{y}_h|c_k)^3 + F_2 p(\mathbf{y}_h|c_k)^2 + F_1 p(\mathbf{y}_h|c_k) + F_0 = 0 \tag{5.2.13}$$

2. Decrease temperature $T \leftarrow \alpha T$

5.2.2 Summary

We have presented a family of EM-based algorithms which solve for the model given in Section 5.1. These algorithms turn out to be quite effective in practice as will be seen in the experimental section. We focus now on a few of the distinguishing features of this approach. First, this is the first model-based attempt at solving the problem of non-redundant clustering. In order to obtain an efficient algorithm for this model, two major assumptions were necessary: the data is binary and Z vectors are obtained according to a mixture of the individual Y features. Second, this is a penalty-based approach in practice. That is, the objective of (5.1.39) seeks to optimize a function which is a tradeoff of a likelihood term and a penalty term (the conditional entropy term.) Thus, the choice of tradeoff weight, β is crucial to determining which solutions are obtained. In some cases, this may be a benefit, as it allows an analyst to weigh how much to penalize redundancies. For instance, if the multiple clusterings in the data set are not orthogonal, a penalty-framework such as this allows one to tune how rigidly the orthogonality constraint is enforced. On the other hand, a penalty-based approach may be unwieldy in practice due to the need for parameter tuning and validation.

Chapter 6

Conditional Information Bottleneck

We now discuss a technique which maximizes a conditional mutual information score. An algorithm is derived using the information bottleneck framework [Tishby et al., 1999] which we term *conditional information bottleneck*. We derive the update equations for the conditional information bottleneck. As stated, the conditional information bottleneck can suffer from mis-coordination among the conditional distributions and so we introduce a coordination term, resulting in the *coordinated conditional information bottleneck*. We then derive algorithms for a variety of distributional assumptions and present experimental results for the algorithms.

6.1 Derivation

We work in a probabilistic setting, where objects are probabilistically assigned to clusters. The goal of data clustering is thus to find a stochastic mapping $P(C|X)$ of objects x to clusters $c \in \{1, \ldots, k\}$, where the number of clusters k is assumed to be given. Here $P(C|X)$ refers to the conditional distribution of C given X, i.e. $p(c|x)$ denotes the probability of assigning object x to cluster c.

6.1.1 Conditional Information Bottleneck (CIB)

Given a particular choice for $P(C|X)$, we would like to quantify the amount of information preserved in the clustering about the relevant features Y. However, we also need to take into account that we assume to have access to the background knowledge Z. A natural quantity to consider is the conditional mutual information $I(C;Y|Z)$ (2.3.5 from Section 2.2.) It describes how much information C, Z convey jointly about relevant features Y compared to the information provided by Z alone. Finding an optimal clustering solution should involve maximizing $I(C;Y|Z)$.

In addition, we would like to avoid over-confidence in grouping objects together. Cluster assignment probabilities should reflect the uncertainty with which objects are assigned to clusters. One way to accomplish this is to explicitly control the fuzziness of the stochastic mapping $P(C|X)$. The latter can be measured by the mutual information $I(C;X)$ between cluster and object identities. Here $I(C;X) = 0$, if objects are assigned to clusters completely at random, whereas $I(C;X)$ becomes maximal for non-stochastic mappings $p(C|X)$. $I(C;X)$ also can be given a well-known interpretation in terms of the channel capacity required for transmitting probabilistic cluster assignments over a communication channel [Tishby et al., 1999].

Combining both aspects, we define the optimal clustering as the solution to the following constrained optimization problem, the Conditional Information Bottleneck (CIB), first introduced in [Gondek and Hofmann, 2003]:

$$\text{(CIB)} \quad p^* = \operatorname*{argmax}_{P(C|X)\in\mathcal{P}} I(C;Y|Z), \quad \text{where}$$

$$\mathcal{P} \equiv \{P(C|X) : I(C;X) \leq C_{\max}\}. \tag{6.1.1}$$

We are looking for probabilistic cluster assignments with a minimal fuzziness such that the relevant information jointly encoded in C, Z is maximal.

In this objective the weight ρ again controls the trade-off between compression (minimizing $I(X,C)$) and preservation (maximizing $I(Y,C|Z)$). Given a certain compression level or channel capacity, the goal is to preserve as much information about Y in C as possible, provided that Z is

revealed to us as side information. Hence, we would like to encode properties of Y in C that cannot be inferred from Z.

For the following derivations it will be useful to again rewrite F in terms of entropies and conditional entropies. We obtain the equivalent objective function

$$F_{CIB} = H(Y|Z) - H(Y|Z,C) - \rho H(C) + \rho H(C|X), \tag{6.1.2}$$

where the $H(Y|Z)$ are constant and so may be safely ignored. We introduce auxiliary variables $q(c)$ and $q(y|z,c)$ and define

$$\tilde{H}(C) = -\sum_c p(c) \log q(c)\,, \tag{6.1.3}$$

$$\tilde{H}(Y|Z,C) = -\sum_{y,z,c} p(y,z,c) \log q(y|z,c)\,, \tag{6.1.4}$$

and an objective

$$\tilde{F}_{CIB} = \tilde{H}(Y|Z,C) - \rho H(C|X) + \rho \tilde{H}(C)\,. \tag{6.1.5}$$

By maximizing the p and q distributions, the objective may be optimized. Using the Lemma in [Tishby et al., 1999], the q parameters which minimize $\tilde{F}_{CIB}$ for given $p(c|x)$ are

$$q(c) = p(c) = \sum_x p(x)p(c|x)\,, \tag{6.1.6}$$

$$q(y|z,c) = p(y|z,c) = \sum_x p(y|x,z)p(x|z,c). \tag{6.1.7}$$

Using the relation

$$\frac{\partial p(y,z,c)}{\partial p(c|x)} = \frac{\partial \sum_x p(x,y,z,c)}{\partial p(c|x)} = p(x,y,z)\,, \tag{6.1.8}$$

the optimal values of the variational parameters are given in terms of the auxiliary parameters as

$$p(c|x) \propto q(c) e^{\frac{1}{\rho} \sum_z p(z|x) \sum_y p(y|x,z) \log q(y|z,c)}\,. \tag{6.1.9}$$

At a stationary point this can again be rewritten more suggestively as a characterization of the resulting joint distribution $p(x,y,z,c)$

$$p(c|x) \propto p(c) e^{-\frac{1}{\rho} \sum_z p(z|x) KL[(p(y|x,z)||p(y|z,c))]}. \tag{6.1.10}$$

A convergence argument identical to the information bottleneck derivation can be applied.

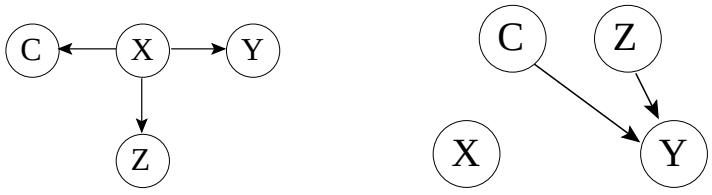
Compared to the Information Bottleneck with Side Information of [Chechik and Tishby, 2002] which uses the objective $\rho I(X,C) - I(Y,C) - \gamma I(Z,C)$, the conditional information bottleneck has the advantage that it alleviates of the need to tune the additional trade-off parameter γ. Notice that tuning γ is problematic, since $I(Y,C)$ and $I(Z,C)$ may live on very different scales, e.g., $I(Y,C)$ may scale with the number of features, while $I(Z,C)$ may scale with the cardinality of the state space of Z. In the conditional information bottleneck, this is taken into account by conditioning on the side information Z in $I(Y,C|Z)$, which enforces non-redundancy without the need for an explicit term $I(Z,C)$ to penalize redundancy.

6.1.2 Derivation from Multivariate Information Bottleneck

The conditional information bottleneck objective may also be derived using the multivariate information bottleneck framework presented in [Friedman et al., 2001]. The multivariate information bottleneck is a technique for characterizing solutions for problems in which there are multiple partitions of the features (observed variables) or parameters (compression variables). Using Bayesian networks specifying the dependencies between these variables, information bottleneck-like objectives may be derived.

Two Bayesian networks, G_{in} and G_{out} must be given, where G_{in} specifies the relation between the observed variables and the compression variables and G_{in} specifies the "relevant" information which is to be preserved. In our problem, which is of a form similar to the parallel information bottleneck described in [Friedman et al., 2001], we obtain the G_{in} pictured in Figure 6.1(a) and the G_{out} pictured in Figure 6.1(b). The network G_{in} specifies that both C and Z compress information about X. The network G_{out} specifies that C and Z should preserve information about Y.

Given networks G_{in} and G_{out}, and using the Multi-Information Bottleneck Principle presented in [Friedman et al., 2001], a Lagrangian which enforces the dependencies specified by G_{in} and G_{out}



(a) G_{in} (b) G_{out}

Figure 6.1: Multivariate Information Bottleneck: Bayesian networks

may be derived. In particular, for our problem this Lagrangian, $\mathcal{L}^{(1)}$, would be

$$\mathcal{L}^{(1)} = I(X;C) + I(X;Z) - \beta I(Y;C,Z). \tag{6.1.11}$$

This can be simplified, first using the fact that $I(X;Z)$ is constant and then expanding the information term:

$$\arg\min \mathcal{L}^{(1)} = \arg\min I(X;C) + I(X;Z) - \beta I(Y;C,Z) \tag{6.1.12}$$

$$= \arg\min I(X;C) - \beta I(Y;C|Z). \tag{6.1.13}$$

where the complete derivation is given in Appendix C.2. Thus, the final result obtained from application of the multivariate information bottleneck recovers our proposed conditional information bottleneck formulation.

As a side note, there are two methods proposed in [Friedman et al., 2001] for obtaining Lagrangians. We have used the first method to derive the conditional information bottleneck formulation from $\mathcal{L}^{(1)}$ as above. The two methods differ in the form of input network G_{out} as well as the focus of the objectives produced. The first, which produces $\mathcal{L}^{(1)}$, focuses on preserving the information in the dependencies of G_{out}, whereas the second which produces $\mathcal{L}^{(2)}$, focuses on preserving information of independencies of G_{out}. The $G^{(b)}_{out}$, which would be used in the second method is pictured in Figure 6.2. This network specifies that X and Y are independent given C, X and Y are independent given Z, and Z and C are independent. The corresponding Lagrangian, $\mathcal{L}^{(2)}$ would be:

$$\mathcal{L}^{(2)} = I(X;C) + I(X;Z) + \gamma\left(I(C;Z) - I(Y;C,Z)\right), \tag{6.1.14}$$

which simplifies as before to yield:

$$\arg\min \mathcal{L}^{(2)} = \arg\min I(X;C) - \gamma \left(I(Y;C|Z) - I(C;Z) \right). \tag{6.1.15}$$

We note that this second formulation $\mathcal{L}^{(2)}$ directly penalizes $I(C;Z)$ whereas the initial formulation $\mathcal{L}^{(1)}$ does not. The $\mathcal{L}^{(1)}$ formulation indirectly penalizes $I(C;Z)$ *through* Y as it selects for C which add information about Y which is not present in Z.

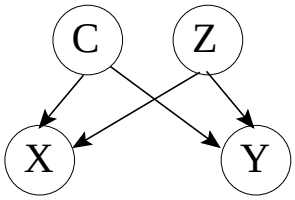


Figure 6.2: $G_{out}^{(b)}$

6.1.3 Coordinated Conditional Information Bottleneck (CCIB)

While the conditional information reflects much of the intuition behind non-redundant data mining, there still is a potential caveat in using (6.1.1): the definition of C may lack global coordination. That is, clustering solutions obtained for different values z may not be in correspondence. For instance, if Z can take a finite number of possible values, then the meaning of each cluster c is relative to a particular value z. The reason for this is that $I(C;Y|Z)$ only measures the information conveyed by C and Z in conjunction, but does not reflect how much relevant information C provides on its own, i.e. without knowing Z. We call this problem the *cluster coordination problem*.

One way to formally illustrate that the CIB does not address the coordination problem is via the following proposition which we state here.

Proposition 1. *Suppose Z and C are finite random variables and define pre-image sets of Z by $\Xi_z = \{x : Z(x) = z\}$. Assume that $P(C|X)$ has been obtained according to* (6.1.1)*. Then one can chose arbitrary permutations π^z over C, one for every value z of Z, and define permuted cluster assignments $p^\pi(c|x) \equiv p^*(\pi^{Z(x)}(c)|x)$ such that $P^\pi(C|X)$ is also optimal for CIB.*

Intuitively this proposition states that by independently renumbering (i.e. permuting) cluster labels within each set Ξ_z, the optimality of the solution is not affected.

A solution to the CIB problem will effectively correspond to a subcategorization or a *local* refinement of the partition induced by Z. Generally, however, one is more interested in concepts or annotations C that are consistent across the whole domain of objects. We propose to address this problem by introducing an additional constraint involving $I(C;Y)$. This yields the following *Coordinated Conditional Information Bottleneck (CCIB)* formulation:

$$\text{(CCIB)} \quad p^* = \underset{P(C|X)\in\mathcal{P}}{\operatorname{argmax}} \; I(C;Y|Z), \quad \text{where} \tag{6.1.16}$$

$$\mathcal{P} \equiv \{P(C|X) : I(C;X) \leq C_{\max}, I(C;Y) \geq I_{\min}\}. \tag{6.1.17}$$

With $I_{min} > 0$ the CCIB favors clustering solutions that obey some global consistency across the sets Ξ_z.

6.1.4 Characterizing solutions

The Lagrangian of the CCIB problem is given by

$$F_{CCIB} = I(C;Y|Z) - \rho I(C;X) + \lambda I(C;Y). \tag{6.1.18}$$

We rewrite mutual informations as (conditional) entropy differences and after dropping constant terms, we arrive at

$$\tilde{F}_{CCIB} = -H(Y|C,Z) - \rho H(C) + \rho H(C|X) - \lambda H(Y|C). \tag{6.1.19}$$

Notice that $H(Y)$ is a constant that can be safely omitted. We introduce cross entropies with auxiliary parameters $q(c)$ and $q(y|c)$, non-negative and normalized, i.e. $\sum_c q(c) = 1$, $\sum_y q(y|c) = 1$ for all c.

We introduce cross entropies with auxiliary parameters $q(c)$, $q(y|c)$, and $q(y|c,z)$, non-negative

and normalized. Now we define

$$\tilde{H}(C) = -\sum_{c} p(c) \log q(c)\,, \tag{6.1.20}$$

$$\tilde{H}(Y|C) = -\sum_{c,y} p(c,y) \log q(y|c)\,, \tag{6.1.21}$$

$$\tilde{H}(Y|C,Z) = -\sum_{c,y,z} p(c,y,z) \log q(y|c,z)\,, \tag{6.1.22}$$

and a new objective

$$\tilde{F}_{CCIB} = -\tilde{H}(Y|Z,C) - \rho\tilde{H}(C) + \rho H(C|X) - \lambda\tilde{H}(Y|C)\,. \tag{6.1.23}$$

The advantage of $\tilde{F}_{CCIB}$ is that it is concave in $P(C|X)$ for given q, since $H(C|X)$ is concave and the $\tilde{H}$ terms are linear in $P(C|X)$. Moreover, $\tilde{F}_{CCIB}$ is also concave in the q parameters for given $P(C|X)$, since the logarithm function is concave.

More precisely the solutions over q can be obtained as follows: First observe that the only $\tilde{H}(C)$ depends on q, only $\tilde{H}(Y|C)$ depends on q, and $\tilde{H}(Y|C,Z)$ on $q(Y|C,Z)$. Differentiating $\tilde{F}_{CCIB}$ with respect to the q parameters and setting to zero results in $q(c) = p(c)$, $q(y|c) = p(y|c)$, and $q(y|c,z) = p(y|c,z)$, as is straightforward to prove (cf. [Tishby et al., 1999], Lemma 2. Proof is given here in Appendix C.1.1.) These correspond to maxima of the function $\tilde{F}_{CCIB}$ for given clustering probabilities $P(C|X)$.

Finally, in order to optimize $\tilde{F}_{CCIB}$ explicitly over the clustering probabilities $P(C|X)$ one makes use of the relation

$$\frac{\partial \sum_x p(c,x,y,z)}{\partial p(c|x)} = p(x,y,z). \tag{6.1.24}$$

and arrives at

$$\frac{1}{p(x)} \frac{\partial \tilde{F}_{CCIB}}{\partial p(c|x)} = [1 + \log q(c|x)] - \log p(c) - \frac{1}{\rho} \sum_{y} p(y|x) \log q(y|c)\,. \tag{6.1.25}$$

Setting to zero and accounting for the normalization of $p(c|x)$ one gets

$$p(c|x) \propto q(c) \exp\left[\frac{\lambda}{\rho} \sum_y p(y|x) \log q(y|c)\right] \times \exp\left[\frac{1}{\rho} \sum_z p(z|x) \sum_y p(y|x,z) \log q(y|c,z)\right]. \tag{6.1.26}$$

At a solution (corresponding to a minimum or saddle-point of $\tilde{F}_{CCIB}$) we can identify q's with p's so Eq. (6.1.27) is equivalent to

$$p(c|x) \propto p(c) e^{-\frac{1}{\rho} KL[(p(y|x)||p(y|c))]}. \tag{6.1.27}$$

which characterizes the true minimum of F_{CCIB}, but potentially other solutions as well.

At a stationary point (corresponding to a maximum or saddle-point of $\tilde{F}_{CCIB}$) this can again be rewritten more suggestively as a characterization of the resulting joint distribution, since there q probabilities agree with their p counterparts. The asymptotic convergence of this scheme is a simple consequence of the fact that $\tilde{F}$ is reduced in every update iteration and that is bounded from below. This may not correspond to a global maximum, but it will fulfill the optimality conditions over the p and q subspaces separately.

6.1.5 Algorithmic approaches

We now discuss algorithmic approaches to obtaining solutions with the derived equations. First as suggested in [Tishby et al., 1999], a deterministic annealing algorithm is derived, and example variants are given for a selection of distributional assumptions. Then, a sequential algorithm is derived based on that given in [Slonim et al., 2002].

6.2 A deterministic annealing algorithm

6.2.1 Categorical side information

The simplest possible case involves side information coming from a classification scheme with small enough cardinality, so that a separate conditional distributions $P(Y|C,Z)$ and $P(Y|Z)$ can be estimated for every combination $(c,z) \in C \times Z$. Then one can simply partition the data into smaller sample sets $\Xi(c,z) = \{y_i : (c_i, y_i, z_i) \in \Xi : c_i = c, z_i = z\}$ and use those sample sets to estimate Y separately for each possible combination $(c,z) \in C \times Z$. We propose to separately parameterize $Q(Y|C)$ and $Q(Y|C,Z)$ by parameters $\tilde{\theta}$ and θ respectively. We can estimate the parameters by conditional maximum likelihood estimation:

$$\begin{aligned} \tilde{\theta}^* &= \operatorname*{argmax}_{\tilde{\theta}} \sum_{i=1}^{n} \log q(y_i|c_i;\tilde{\theta}), \quad \text{and} \\ \theta^* &= \operatorname*{argmax}_{\theta} \sum_{i=1}^{n} \log q(y_i|c_i,z_i;\theta). \end{aligned} \tag{6.2.1}$$

Hence the estimation equations for the Q distributions become in the Gaussian case

$$\theta_{c,z} = \frac{\sum_{y \in \Xi(c,z)} y}{|\Xi(c,z)|}, \quad \tilde{\theta}_z = \frac{\sum_{c \in C} \sum_{y \in \Xi(c,z)} y}{\sum_{c \in C} |\Xi(c,z)|} \tag{6.2.2}$$

which is simply the sample mean. In case of binary variables Y, we can use a Bernoulli sampling model with success probabilities $\theta_{c,z}$, $\tilde{\theta}_c$ and get the same maximum-likelihood equations (6.2.2). We will now discuss a number of special cases to illustrate the usefulness of the CCIB approach.

Multinomial features

We consider the case where features Y are distributed according to a multinomial distribution. We propose to use the empirical feature counts or maximum likelihood estimates as a plug-in estimator for $P(Y|X)$ and a uniform distribution over documents, which produces the iterative update equations of Algorithm 6.1.

Algorithm 6.1 Update equations for multinomial Y, categorical Z

Initialized data probabilities

$$p(x) = 1/n \tag{6.2.3}$$
$$p(z|x) = \delta(z, z(x)) \tag{6.2.4}$$
$$p(y_j|x, z) = p(y_j|x) = \frac{n_j(x)}{\|n(x)\|_1} \tag{6.2.5}$$

Iterative update equations

Auxiliary equations:

$$q(c) = \frac{1}{n} \sum_x p(c|x) \tag{6.2.6}$$
$$q(c, z) = \frac{1}{n} \sum_{x:z(x)=z} p(c|x) \tag{6.2.7}$$
$$q(c, y_j) = \frac{1}{\sum_x \|n(x)\|_1} \sum_x n_j(x) p(c|x) \tag{6.2.8}$$
$$q(c, y_j, z) = \frac{1}{\sum_x \|n(x)\|_1} \sum_{x:z(x)=z} n_j(x) p(c|x) \tag{6.2.9}$$

Membership equation:

$$p(c|x) \propto q(c) e^{\frac{1}{\rho} \sum_{j=1}^m \frac{n_j(x)}{\|n(x)\|_1} \log \frac{q(c, y_j, z(x))}{q(z(x), c)}} \tag{6.2.10}$$

Binary features with Naive Bayes assumption

Let us now consider the case in which $Y = \{0, 1\}^d$ is a binary vector. We make the Naive Bayes assumption and model each feature as a Bernoulli trial. As in the previous case we assume a uniform distribution over documents and use the maximum likelihood estimates for $P(Y|X)$. The update equations in this case are presented in Algorithm 6.2.

Gaussian features

We also consider the case that Y is generated by Gaussian. In particular, assume Y is vector-valued, distributed according to a multivariate Gaussian distribution with covariance matrix $\sigma^2 I$ for each s, z. The resulting update equations are shown in Algorithm 6.3.

Algorithm 6.2 Update equations for binary Y, categorical Z

Initialized data probabilities

$$p(x) = 1/n \tag{6.2.11}$$
$$p(z|x) = \delta(z, z(x)) \tag{6.2.12}$$
$$p(y_j|x, z) = p(y_j|x) = b_j(x) \tag{6.2.13}$$

Iterative update equations

Auxiliary equations:

$$q(c) = \frac{1}{n} \sum_x p(c|x) \tag{6.2.14}$$
$$q(c, z) = \frac{1}{n} \sum_{x:z(x)=z} p(c|x) \tag{6.2.15}$$
$$q(c, y_j) = \frac{1}{n} \sum_x b_j(x) p(c|x) \tag{6.2.16}$$
$$q(c, y_j, z) = \frac{1}{n} \sum_{x:z(x)=z} b_j(x) p(c|x) \tag{6.2.17}$$

Membership equation:

$$\begin{aligned} p(c|x) \propto & q(c) e^{\frac{1}{\rho} \sum_{j=1}^{m} \left(b_j(x) \log \frac{q(c, y_j, z(x))}{q(c, z(x))} \right)} \\ & \cdot e^{\frac{1}{\rho} \sum_{j=1}^{m} \left((1 - b_j(x)) \log \left(1 - \frac{q(c, y_j, z(x))}{q(c, z(x))} \right) \right)} \\ & e^{\frac{1}{\rho} \sum_{j=1}^{m} \left(b_j(x) \log \frac{q(c, y_j)}{q(c)} \right)} \\ & \cdot e^{\frac{1}{\rho} \sum_{j=1}^{m} \left((1 - b_j(x)) \log \left(1 - \frac{q(c, y_j)}{q(c)} \right) \right)} \end{aligned} \tag{6.2.18}$$

6.2.2 Continuous-valued known structure

We next consider the case, where Z is real-valued vector, e.g. each component of Z might be a function of Y or in the simplest case a subset of the components of Y. Then, in order to estimate $q(Y|C, Z)$ one has to fit a regression model that predicts Y from Z for every cluster C. A generalized linear model may be used to estimate $q(y|c, z; \theta)$. In practice, we fit predictors for each response variable Y_j independently. For each feature j we create K models conditioned on C. The vector $Z(x)$ is used as the observation for each instance $x \in C$ in order to create the data matrix for a predictor. As assignments may be soft, for a given c we include all instances with non-zero probability $p(c|x)$ and use the membership probabilities $p(c|x)$ as prior weights. Any of a number of link functions

Algorithm 6.3 Update equations for Gaussian Y, categorical Z

Initialized data probabilities

$$p(x) = 1/n \tag{6.2.19}$$
$$p(z|x) = \delta(z, z(x)) \tag{6.2.20}$$

Iterative update equations

Auxiliary equations:

$$q(c) = \frac{1}{n} \sum_x p(c|x) \tag{6.2.21}$$
$$q_{CZ}(c, z) = \frac{1}{n} \sum_{x:z(x)=z} p(c|x) \tag{6.2.22}$$
$$\theta_c = \frac{\sum_{x:c(x)=c} Y(x)}{\sum_{x:c(x)=c} 1}, \theta_{c,z} = \frac{\sum_{x:c(x)=c,z(x)=z} Y(x)}{\sum_{x:c(x)=c,z(x)=z} 1} \tag{6.2.23}$$
$$q(y|z) = \exp\Big[-\frac{1}{2\sigma^2} \|\theta_c - y\|^2 + const \Big] \tag{6.2.24}$$
$$q(y|c, z) = \exp\Big[-\frac{1}{2\sigma^2} \|\theta_{c,z} - y\|^2 + const \Big] \tag{6.2.25}$$

Membership equation:

$$p(c|x) \propto q(c) e^{-\frac{1}{\rho}(\|\theta_{c,z} - Y(x)\|^2 + \lambda \|\theta_c - Y(x)\|^2)} \tag{6.2.26}$$

h may be used; among these are the canonical link functions for normal, Poisson, and binomial distributions. The update equations are given in Algorithm 6.4.

6.2.3 Optimization algorithm

The iterative algorithms presented thusfar optimize the objective for a given ρ. We now build on these algorithms to produce hard-clustering algorithms. A number of hard-clustering versions of the information bottleneck exist. Among these are the deterministic annealing Information Bottleneck first suggested in [Tishby et al., 1999] for soft clustering and the sequential Information Bottleneck [Chechik and Tishby, 2002] for hard clustering. With application to large data sets in mind, we consider a simple deterministic annealing scheme with fixed K.

Algorithm 6.4 Update equations for Y distribution chosen via GLM link function h, non-categorical Z

Initialize data probabilities

$$p(x) = 1/n \tag{6.2.27}$$

Iterative update equations

Update auxiliary equations:

$$q(c) = \frac{1}{n} \sum_x p(c|x) \tag{6.2.28}$$

Update GLM

$$\begin{aligned} &\forall j = 1 \dots |Y| : \\ &\qquad \theta_j = \text{GLM-FIT}_h(Z, Y_j, p(C|X)) \end{aligned} \tag{6.2.29}$$

Membership equation:

$$p(c|x) \propto q(c) e^{\frac{1}{\rho} q(y|c,z;\theta)} \tag{6.2.30}$$

A deterministic annealing algorithm for non-redundant clustering

Continuation methods are a successful approach to performing clustering [Rose, 1998]. We present a simple deterministic annealing-based algorithm for fixed K. The algorithm begins with inverse temperature $\beta = 1/\rho$ set to 0 and increases β by a multiplicative factor $b > 1$ until a hard-clustering solution is obtained. For low β (high temperature), the membership probabilities $p(c|x)$ will be approximately equal. As the β is increased, which corresponds to lowering the temperature, a hard-clustering solution will result.

Algorithm 6.5 Deterministic Annealing Algorithm for Non-redundant Clustering

initialize inverse temperature $\beta = \frac{1}{\rho} = 0$
initialize $p(c|x)$ randomly
repeat
  iterate equations from Algorithm 6.1, 6.2, 6.3, or 6.4 until converged
  increase inverse temperature: $\beta = b \cdot \beta$
until no change in $p(c|x)$ and $\forall c, x : p(c|x) \in \{0, 1\}$

6.3 Hard clustering algorithms

For a technique which performs the clustering entirely assuming hard assignments, we present variants of the *sequential Information Bottleneck (sIB)* method, described in [Slonim et al., 2002]. Though the techniques differ in the framework by which clusters are initialized and merged, they share the same merging update and criteria equations which we derive for the conditional case below.

6.3.1 Hard Clustering

Taking the limit $\beta \to \infty$ for the self-consistent equations result in the following equations for the hard cluster case.

$$p(c|x) = \begin{cases} 1 & \text{if } x \in c \\ \\ 0 & \text{otherwise} \end{cases}, \tag{6.3.1}$$

$$p(c) = \sum_{x \in c} p(x), \tag{6.3.2}$$

$$p(y, c, z) = \sum_{x \in c} p(x)p(y|x)p(x|z), \tag{6.3.3}$$

$$p(c, z) = \sum_{x \in c} p(x)p(x|z). \tag{6.3.4}$$

6.3.2 Merger equations

Lemma 6. *In the case of hard clustering, the probabilities from any merger $(c_i, c_j) \Rightarrow \bar{c}$ are defined by the following equations,*

$$p(\bar{c}) = p(c_i) + p(c_j), \tag{6.3.5}$$

$$p(\bar{c}, z) = p(c_i, z) + p(c_j, z), \tag{6.3.6}$$

$$p(y, \bar{c}, z) = p(y, c_i, z) + p(y, c_j, z), \tag{6.3.7}$$

where

$$\Pi = \{\pi_i, \pi_j\} = \left\{ \frac{p(c_i)}{p(\bar{c})}, \frac{p(c_j)}{p(\bar{c})} \right\}, \tag{6.3.8}$$

$$\Pi(z) = \{\pi_i(z), \pi_j(z)\} = \left\{ \frac{p(c_i, z)}{p(\bar{c}, z)}, \frac{p(c_j, z)}{p(\bar{c}, z)} \right\}. \tag{6.3.9}$$

Algorithm 6.6 Sequential method

Input: X objects to be clustered, parameter K

Output: C_m: m-partition of X into K clusters

Initialization:

- Construct $C \equiv X$:
  - $C \leftarrow$ random partition of X into K clusters
  - For c_i in C:
    - $p(c_i) = \sum_{x_j \in c_i} p(x_j)$
    - $p(y, c_i, z) = \sum_{x_j \in c_i} p(x_j) p(y|x_j) p(z|x_j) \quad \forall y \in Y, z \in Z$
    - $p(c_i, z) = \sum_{x_j \in c_i} p(x_j) p(z|x_j) \quad \forall z \in Z$
    - $p(c_i|x_j) = 1$ if $x_j \in c_i$, 0 otherwise

Loop:

- For $t = 1 \ldots (N-1)$
  - randomly draw x_j to $c_{from}(x_j)$
  - $c'_{from} = c_{from} \backslash \{x_j\}$
  - $c_{to} = \arg\min_{c_i} d(x_j, c_i)$
  - $c'_{to} = c_{to} \cup \{x_j\}$
    - $p(c'_{to}) = p(c_{to}) + p(x_j)$
    - $p(c'_{from}) = p(c_{from}) - p(x_j)$
    - $p(c'_{to}|x) = 1$
    - $p(c'_{from}|x) = 0$
    - $p(c'_{to}, z) = p(c_{to}|z) + p(x_j|z) \quad \forall y \in Y$
    - $p(c'_{from}, z) = p(c_{from}|z) - p(x_j|z) \quad \forall y \in Y$
    - $p(y, c'_{to}, z) = p(y, c_{to}, z) + p(y, x_j, z) \quad \forall y \in Y, z \in Z$
    - $p(y, c'_{from}, z) = p(y|c_{from}, z) + p(y, x_j, z) \quad \forall y \in Y, z \in Z$
  - Replace c_{from} with c'_{from}, c_{to} with c'_{to}

6.3.3 Merging criteria

Both methods select which two clusters to merge based on the difference in $\mathcal{L}$ before and after the merge.

For values L^{bef} before the merge and L^{aft} after the merge, the *merger cost* which we wish to minimize, is defined as $\Delta\mathcal{L}(c_i, c_j) = \mathcal{L}^{aft} - \mathcal{L}^{bef}$, which in our case is:

$$\Delta\mathcal{L}(c_i, c_j) = \mathcal{L}^{aft} - \mathcal{L}^{bef} \tag{6.3.10}$$

$$= I(X; C^{aft}) + \beta H(Y|C^{aft}, Z) - (I(X; C^{bef}) + \beta H(Y|C^{bef}, Z)). \tag{6.3.11}$$

Lemma 7. *The merger cost $\Delta\mathcal{L}(c_i, c_j)$ for merge $(c_i, c_j) \Rightarrow \bar{c}$ is given by:*

$$\Delta\mathcal{L}(c_i, c_j) = -H(X|C = \bar{c}) + H(X|C = c_i) + H(X|C = c_j) + \beta\left[H(Y|C = \bar{c}, z) - H(Y|C = c_i, z) - H(Y|C = c_j, z)\right] \tag{6.3.12}$$

and may be rewritten:

$$\Delta\mathcal{L}(c_i, c_j) = -p(\bar{c}) \cdot JS_{\Pi}[p(x|c_i), p(x|c_j)] + \beta \sum_z p(\bar{c}|z) \cdot JS_{\Pi(z)}[p(y|c_i, z)||p(y|c_j, z)]. \tag{6.3.13}$$

which in the case of hard clustering, simplifies to:

$$\Delta\mathcal{L}(c_i, c_j) = -p(\bar{c})H(\Pi) - \beta \sum_z p(\bar{c}, z) \cdot JS_{\Pi(z)}\left[p(y|c_i, z) \| p(y|c_j, z)\right]]. \tag{6.3.14}$$

6.3.4 Algorithm

The sequential algorithm is presented in Figure 6.6. It begins by randomly initializing hard assignments. It then loops through all instances making local improvements until no improvement is possible. Local improvements consist of reassigning a point to the cluster which optimizes the objective function.

6.4 Summary

The coordinated conditional information bottleneck (CCIB) uses a conditional mutual information score to factor out the known structure. This approach captures the natural tradeoff between

clustering quality and non-redundancy. While there is an additional parameter necessary to ensure coordination of solutions, empirically the algorithm is insensitive to this. The technique may be applied to a variety of statistical models and categorical or, with the use of generalized linear models, continuous known structure. Both deterministic annealing and sequential clustering algorithms may be easily derived.

Chapter 7

Conditional Ensembles

We now present an algorithm for non-redundant clustering which makes use of *cluster ensembles*. One chief advantage of this framework is that virtually any clustering algorithm: k-means, EM, k-nearest-neighbors, etc. may be used. Unlike the Coordinated Conditional Information Bottleneck of the previous section, global coordination is not explicitly enforced and so we can make no theoretical guarantees that a coordinated objective is maximized. Instead, we investigate theoretically what pre-conditions are necessary for the desired behavior.

7.1 Algorithm

We will begin by providing an abstract, high-level view of the proposed algorithm, which illustrates the basic idea. As shown in Figure 7.1 the algorithm – which we call *CondEns* for Conditional Ensemble clustering – operates in three stages. In the first stage, local clustering solutions are computed for all data points belonging to the same cluster of the given clustering. This results in one refining clustering for every cluster of the original clustering. The second stage extends these local solutions to create global clustering solutions. How this is done depends on the specific base clustering method employed. Note that this stage is meant not to change the identity of clusters.

Algorithm 7.1 *CondEns* Algorithm

Input: Training data $\{x_1, \ldots, x_n\}$, clustering $Z : \{1, \ldots, n\} \to \{1, \ldots, l\}$, number of clusters k in target clustering, number of clusters k_j for each local clustering

Output: Clustering $C : \{1, \ldots, n\} \to \{1, \ldots, k\}$

1. Clustering

Partition data set into pre-image sets:
$I_j(Z) \equiv \{i : Z(i) = j\}$ and $X_j(Z) \equiv \{x_i : i \in I_j(Z)\}$, $j = 1 \ldots, l$.

Apply base clustering method to each $X_j(Z)$ to find a local clustering
$\tilde{C}^j : I_j(Z) \to \{1, \ldots, k_j\}$, $j = 1, \ldots, l$.

2. Extension

Extend each $\tilde{C}^j$ to a global clustering C^j by assigning training instances in $X_m(Z)$, $m \neq j$ to one of the existing clusters
$C^j : \{1, \ldots, n\} \to \{1, \ldots, k_j\}$, $j = 1 \ldots, l$

3. Combination

Combine clustering solutions C^j to form a consensus clustering:
$C = \mathsf{Consens}(C^1, \ldots, C^l)$, where $C : \{1, \ldots, n\} \to \{1, \ldots, k\}$

That is, in model-based or centroid-based approaches one would keep the model parameters or cluster centroids constant. For instance, with Expectation Maximization (EM), this stage corresponds to application of the E-step over all instances. With k-means, this stage corresponds to application of the assignment step over all instances. Finally, the global clustering solutions obtained from the local seeds are combined using an ensemble clustering method to produce the final coordinated clustering.

The rationale behind the algorithm is that by clustering within each group of the given partitioning, i.e. by computing local refinements, one will effectively avoid reproducing the given solution. In addition, it seems intuitive that an alternative clustering, that is in some sense orthogonal to the existing one, should also show up in each of the groups induced by the latter. If this intuition is correct, then the combination step should be able to recover it.

There are two questions that remain to be settled. First, one needs to render the combination step more precisely. This involves the specification of a concrete combination scheme for the consensus clustering stage. Second, it would be reassuring if the above intuitions could be underpinned with theoretical analysis. We present such an attempt to shed some light on the working and assumptions

of the above algorithm in Section 7.3.

7.2 Ensemble Clustering

Ensemble clustering was reviewed in Section 3.6. In our case, we are only concerned with the combination step and not with the problem of how to generate the different clustering solutions which are obtained via the first two steps of the *CondEns* algorithm.

To restate the procedure in this setting, we are given l clusterings $C = C^1, \ldots, C^l$ obtained from steps 1 and 2 in 7.1 where each clustering C^j partitions the data into k clusters. The goal is then to find a combined partition C of k clusters.

Following [Topchy et al., 2003] we have obtained good results using the quadratic mutual information criterion:

$$C = \underset{C'}{\operatorname{argmax}} \sum_{j=1}^{l} I^2(C'; C^j). \tag{7.2.1}$$

This criterion is up to a factor of 2 equivalent to the *category utility function* U presented in [Gluck and Corter, 1985]. Moreover, it was shown in [Mirkin, 2001] that maximization of 7.2.1 for a fixed number of target clusters is equivalent to the minimization of a squared-error criterion. The latter can be solved using standard approximation techniques such as a k-means clustering. This is the approach we will apply in the experiments.

7.3 Analysis

7.3.1 Complexity

The technique presented has attractive overall time complexity. For instance, in the case where k-means is used for the base clustering method, *CondEns* is typically faster than running traditional k-means over the data set. This surprising result is due to the fact that *CondEns* first partitions the data set into smaller pre-image sets. As the complexity of k- means clustering is typically superlinear

in the number of instances[Bottou and Bengio, 1995, Davidson and Satyanarayana, 2003], and the Extension and Combination steps of *CondEns* are relatively fast, the divide-and-conquer approach of *CondEns* achieves time savings over traditional k-means.

More precisely, the K-means clustering algorithm (for K clusters) has complexity $O(IKND)$ where K is the number of clusters, N the number of instances, D the dimensionality of the feature space, and I is the number of iterations required for convergence. As the number of iterations necessary for convergence on a data set, I, is not known in advance and may vary widely depending on the data set and random initialization, the $O(iKND)$ may not be precisely computed but is still useful for comparative purposes.

Assuming the data is partitioned among the ensemble equally, the initial clustering by base clustering method performed in Step 1 has complexity $O(I_1 K \frac{N}{M} D)$. Extension in the Step 2 may be done in $O(KMND)$. The final consensus combination in Step 3 would have complexity $O(I_3 KNM)$, yielding a total of $O(I_1 K \frac{N}{M} D + MKND + I_3 KNM)$. This can be compared to running traditional K-means on the data set, yielding a complexity of $O(IKND)$. Furthermore, we may apply the results of [Davidson and Satyanarayana, 2003], which argues that under certain assumptions, $I = \alpha N$ for constant α, and establishes experimentally that this relationship holds across a wide variety of data sets. Using this conclusion, we can refine the complexities further, giving a total complexity for *CondEns* of $O(K \frac{N^2}{M^2} D + MKND + KN^2 M)$ versus that of K-means, $O(KN^2 D)$. In our setting, M, the number of pre-image sets, will be considerably smaller than the dimensionality of the data: $M << D$, which ensures a complexity competitive with conventional K-means.

7.3.2 Conditions for correctness

In this section, we will provide a partial analysis of when a certain target clustering embedded in the data set can be recovered by the above algorithm. The analysis requires making a number of assumptions on the base clustering algorithm, the cluster combination scheme, and the relative dominance of the target clustering. Certainly, these requirements are quite strong and we do not

claim that they can be met in most realistic settings. However, we believe they provide at least some strong motivation for the proposed method and that the insight gained by the analysis is valuable in supplementing the empirical evidence presented in Section 8.

A crucial quantity to review in the current setting is the conditional mutual information, defined as

$$I(A;B|C) = H(A|C) - H(A|B,C)\,, \tag{7.3.1}$$

where A, B, and C are random variables. We state a few useful facts about conditional mutual information that directly follow from this definition.

Proposition 2.

(a) $I(A;B|C) = I(B;A|C)$.

(b) $I(A;B|C) = I(A;B,C) - I(A;C)$.

(c) $I(A;B|C) - I(A;B) = I(A;C|B) - I(A;C)$.

(d) $A \perp\!\!\!\perp C|B$ *if and only if* $I(A;C|B) = 0$.

We will quantify the quality of a clustering as the mutual information between the feature representation and the cluster membership variable, i.e.

$$Q(C) \equiv I(C;X) = H(X) - H(X|C)\,, \tag{7.3.2}$$

where H denotes the (conditional) entropy or the differential (conditional) entropy, dependent on the type of feature representation. Herein we restrict clusterings to be deterministic functions of the feature representation, i.e. $C : X \to \{1,\ldots,k\}$. We will write X for the input random variable and $C(X)$ for the induced random variable over clusters. A simple consequence is that C will be conditionally independent of any other random variable, given the input. In particular C is independent of any given partitioning $Z = Z(X)$, i.e. $C \perp\!\!\!\perp Z|X$ or, equivalently $I(C;Z|X) = 0$, $\forall\, C$. Since we are interested in clusterings that are in some sense orthogonal to an existing clustering, we need to clarify the meaning of this concept. We proceed in the spirit of [Dom, 2002] which

proposes a mutual-information score for cluster validity and [Meilă, 2003] which argues for the use of a mutual-information score as it provides a true metric on the space of clusterings.

Definition 16. *C and Z are information-orthogonal (w.r.t. X), if*

$$I(C, Z; X) = I(C; X) + I(Z; X). \tag{7.3.3}$$

This condition is a natural choice as it can be shown to be equivalent to the condition that C and Z are independent as well as to the condition that, using the metric of [Meilă, 2003], C and Z attain the maximal distance possible. We note a first consequence.

Lemma 8. *If C and Z are information-orthogonal, then $I(C; X) = I(C; X|Z)$ and $I(Z; X) = I(Z; X|C)$.*

Proof. Using the chain rule for mutual information,

$$I(C, Z; X) = I(C; X|Z) + I(Z; X) \tag{7.3.4}$$

$$= I(Z; X|C) + I(C; X), \tag{7.3.5}$$

which when combined with (7.3.3) establishes that $I(C; X) = I(C; X|Z)$ and $I(Z; X) = I(Z; X|C)$. □

For arbitrary clusterings C, Lemma 8 may not hold, however, the conditional independence from Z given X guarantees the following:

Lemma 9. $I(C; X|Z) \leq I(C; X)$

Proof. Since $I(C; Z|X) = 0$ one gets, $I(C; X|Z) = I(C; Z|X) + I(C; X) - I(C; Z) = I(C; X) - I(C; Z) \leq I(C; X)$. □

An immediate implication of the two lemmata is the following corollary.

Corollary 1. *If $C^* = \operatorname{argmax}_C I(C; X)$ and C^* is information-orthogonal to Z, then $C^* = \operatorname{argmax}_C I(C; X|Z)$.*

Proof. $I(C^*; X|Z) = I(C^*; X) \geq I(C; X) \geq I(C; X|Z)$ for all C. □

The main problem is that the bound in Lemma 9 holds only on the average. Namely, one may have a situation where $I(C'; X|Z = z_j) > I(C; X|Z = z_j)$, despite the fact that $I(C'; X|Z) \leq I(C; X|Z)$. While a general analysis seems difficult, we can prove that a dominant, information-orthogonal clustering will also dominate all other clusterings in at least one group of Z. We will require the following lemma:

Lemma 10. *There exists a C where:*

$$I(C; X|Z) = \sum_j p(z_j) I(C^j; X|Z = z_j). \tag{7.3.6}$$

This lemma is proven by using the fact that Z is a partition, only the X^j participate in $I(C^j; X|Z = z_j)$ and so in C each X^j may be handled separately. For a detailed derivation, see Appendix D.1.

Proposition 3. *If C^* is information-orthogonal to Z and $I(C^*; X) > I(C; X)$ for all $C \in \mathcal{C} - \{C^*\}$, where $\mathcal{C}$ is chosen such that for all $C^j \equiv \text{argmax}_C\, I(C; X|Z = z_j)$, $C^j \in \mathcal{C}$, then there exists at least one group j^* such that $I(C^*; X|Z = z_{j^*}) \geq I(C; X|Z = z_{j^*})$ for all C.*

Proof. We first point out that there exists some C such that $I(C; X|Z) = \sum_j p(z_j) I(C^j; X|Z = z_j)$. Such a C is constructed such that the restriction of C to the pre-images X^j of Z equals C^j. Using the fact that Z is a partition, only those X^j participate in $I(C^j; X|Z = z_j)$ and so in C each X^j may be separately assigned to its C^j. In conjunction with Lemma 9 this yields:

$$\sum_{j=1}^{l} p(z_j) I(C^j; X|Z = z_j) = I(C; X|Z) \leq I(C; X).$$

By assumption $I(C; X) < I(C^*; X)$ and with Lemma 8 one arrives at

$$I(C; X) < I(C^*; X) = I(C^*; X|Z) = \sum_{j=1}^{l} p(z_j) I(C^*; X|Z = z_j). \tag{7.3.7}$$

If one now assumes that all $C^j \neq C^*$, then by the local optimality of C^j one would get $I(C^j; X|Z = z_j) \geq I(C^*; X|Z = z_j)$ which contradicts the fact that

$$\sum_{j=1}^{l} p(z_j) I(C^j; X|Z = z_j) < \sum_{j=1}^{l} p(z_j) I(C^*; X|Z = z_j)$$

because of the non-negativity of the individual terms in the sum. Hence there has to be at least one index j^* so that C^* is optimal for $Z = z_{j^*}$. □

The relevance of this proposition is that asymptotically, if the base cluster method uses the mutual information criterion to derive local clustering solutions and if the target clustering is dominant and information-orthogonal to the given clustering, then the target clustering will be among the clustering solutions handed to the combination stage. One could hence select a clustering solution among the l clusterings C^j. However, in practice a meaningful combination procedure is a far more useful alternative.

Chapter 8

Experimental Results

8.1 Experimental design

8.1.1 Data sets with multiple clusterings

We consider synthetic and real-world data sets which contain multiple clusterings. Experiments are performed where one of the clusterings is assumed known, and the algorithms are evaluated according to ability to discover a remaining clustering. We begin by discussing the criteria used for evaluation and then present the experimental results.

8.1.2 Evaluation criteria

Evaluating the success of clustering algorithms requires a method by which to assess the quality of clusterings obtained. The evaluation of clusterings is sometimes referred to as *cluster validity* and has a rich literature [Jain and Dubes, 1988, Bezdek and Pal, 1998, Halkidi et al., 2001]. In [Jain and Dubes, 1988], indices of cluster validity are classified into three approaches: internal criteria, external criteria, and relative criteria, which are defined below. The distinction is made

between cluster validation *criteria*, which express approaches for cluster validation and cluster validation *indices*, which are specific statistics computed within these approaches. This is an important distinction as some cluster validation indices may be used for more than one criteria.

External criteria measure the quality of a clustering by comparing it to a known clustering of the data. Typically the known clustering corresponds to some categorization scheme known a priori.

Internal criteria measure the quality of the clustering using only the data. Typical criteria favor clusterings with high similarity within clusters and low similarity across clusters. For instance, if an explicit objective function exists, it may be used as an internal index.

Relative criteria compares the quality of two clusterings. Typically, such criteria are used to decide on the best fit from a set of clusterings produced from an algorithm using different settings. "Best fit" may correspond to the most stable solution or it may correspond to the highest quality solution which is often determined using indices associated with external or internal criteria.

In this work, we will consider data sets where existing categorizations are known and so can make use of external criteria. We prefer to use an external criteria approach as it can be applied to real-world data sets and give better indication of how the algorithms would perform in real settings.

8.1.3 External indices

We briefly review standard external indices and discuss the particular indices used in the experiments.

Precision-based measures

A standard measure used in classification is *precision* which measures the proportion of instances which were classified correctly. In particular, the *micro-averaged precision* is defined as:

Micro-averaged precision The result of the algorithm C and ground-truth classification L are given. For $\alpha(l, c)$, the number of instances correctly assigned class label l and $\beta(l, c)$, the number of instances incorrectly assigned class label l, the micro-averaged precision is computed as:

$$Prec_L(C) = \frac{\sum_c \alpha(l, c)}{\sum_c \alpha(l, c) + \beta(l, c)}. \tag{8.1.1}$$

The precision may not be used as stated, however, as it assumes a correspondence between the labels of L and the labels of C. In clustering, the labels are arbitrary and so a meaningful mapping from the labels of C to the labels of L must be found. In work such as [Slonim et al., 2002], all instances of a cluster are assigned to the most prevalent ground-truth label in that cluster. We instead use the Hungarian Method [Kuhn, 1955] in order to obtain an optimal correspondence between the clusters found by the algorithm and the class labels of the ground-truth classification. Once such a correspondence is found, standard metrics such as precision may be used for evaluation.

One important restriction of using precision is that the number of labels in C and L are assumed to be equal. Clearly in practice, the number of clusters K may not equal the number of classes in a ground-truth classification scheme. For this problem in particular, where there may be multiple classification schemes for a single data set, the multiple classification schemes may not share the same cardinality. In order to compare the similarity of a clustering to the existing classification schemes, a measure is needed which does not require the cardinalities be equal.

Pair-counting indices

A number of commonly used cluster validity indices which do not have cardinality constraints can be defined in terms of counts of pairs of instances[Hubert and Schultz, 1976], [Jaccard, 1912], [Rand, 1971], [Fowlkes and Mallows, 1983]. As discussed in [Meilă, 2003], these indices suffer from a number of drawbacks. In order to compare clusterings with these indices, they must be normalized. Normalization typically requires determining the baseline, the minimum attainable value for the index. One drawback is that this baseline has to be recomputed for each clustering. Further, the

value for the baseline varies sharply for many clusterings making it questionable to compare scores with different baselines. Finally, the baseline is the expectation for the null hypothesis, however the null hypothesis assumes the two clusterings are sampled independently and are sampled for fixed cluster sizes. The algorithms typically considered do not preserve cluster sizes between runs, and so these assumptions are violated in practice.

Normalized Mutual Information

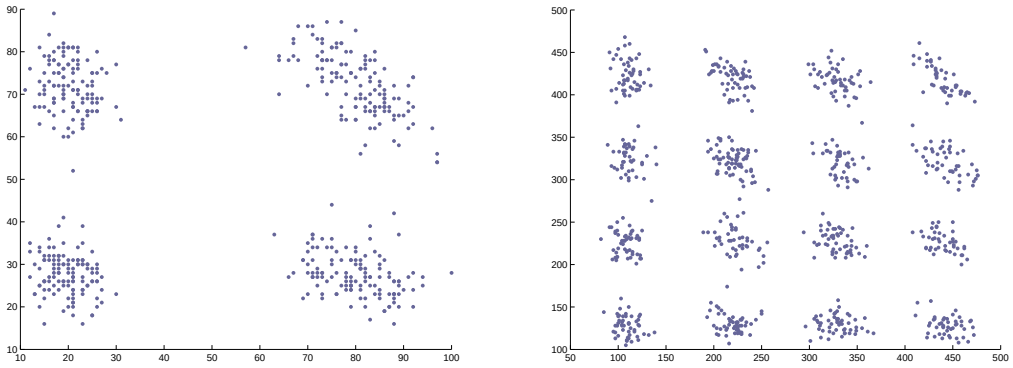
The mutual information $I(C;L)$ provides a natural approach to measuring how much information about L is captured by C, and has been used for cluster validation [Vaithyanathan and Dom, 1999b]. As $I(C;L)$ may vary in scale, we will use the *normalized mutual information*:

Normalized mutual information (NMI) Normalized mutual information uses the upper bound on $I(C;L)$, which is at most $H(L)$. Thus the mutual information $I(C;L)$ may be normalized:

$$NMI(C) = \frac{I(C;L)}{H(L)}. \tag{8.1.2}$$

8.2 Experiments

In this section we present results on synthetic and real data sets, which allows a comparison of the algorithmic techniques discussed. We are primarily concerned with the ability of the algorithms to obtain underlying structure which is non-redundant with respect to known structure, however we are also interested in characterizing the relative computational efficiency of the algorithms as well as their sensitivity to initialization and parameter choices.



(a) k=2 (b) k=4

Figure 8.1: Sample 2-d synthetic sets generated according to the description in Section 8.3.1 using $\alpha = 1$ and parameter sets: {k=2, n=200, m=400}, {k=4,n=800,m=1100}.

8.3 Synthetic data sets: Revealing secondary structure

8.3.1 2-D synthetic sets

Generation

In order to study the robustness of the algorithm with respect to the orthogonality assumption, two-dimensional synthetic data sets with two clusterings A and B are generated using a multinomial model. For the first dimension, which is associated with clustering A, for each of the clusters $p_1, \ldots p_K$, feature probabilities $\theta_{p1} \ldots \theta_{pK}$ are chosen to be equally spaced on the interval [0,1]. The feature probabilities for B are obtained by first taking $\theta'_{q1}, \ldots \theta'_{qK}$ equally spaced on the interval [0,1]. Next, these probabilities are associated with the θ_{pk} according to an orthogonality weight, α in the following expression: $\theta_{qk} = \alpha\theta_{qk} + (1 - \alpha)\theta_{pk}$. An instance is sampled by randomly selecting membership in $\{p_1, \ldots, p_K\}$ and $\{q_1, \ldots, q_K\}$ and sampling from the associated distributions. At $\alpha = 1$, A and B are orthogonal and as $\alpha \to 0$, the clusterings A and B become perfectly correlated.

Table 8.1: Results for 2d Synthetic Sets: 50 randomly generated data sets with K=2, N=200, M=400. Clustering A is assumed to be known and algorithms are evaluated on ability to find B. Algorithms are run with 10 random initializations for each data set. Results are shown for the mean, max and min precision over initializations, averaged over all 50 sets. Highest scores are in bold.

$K = 2$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.5299	0.5299	0.5299	**1.0000**	**1.0000**	**1.0000**	0.16s
daCCIB	0.5424	0.6285	0.5157	0.8470	**1.0000**	0.5396	1.25s
CondEns[k-means]	0.7403	0.7628	0.7202	0.7643	0.7843	0.7417	0.02s
CondEns[EM]	0.5933	0.6396	0.5395	0.9145	0.9734	0.8636	0.02s
CondEns[daIB]	0.5900	0.6426	0.5359	0.9173	0.9746	0.8619	0.25s

Table 8.2: Results for 2d Synthetic Sets: 50 randomly generated data sets with K=3, N=450, M=700.

$K = 3$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.5179	0.5874	0.3935	0.6553	0.8486	0.4881	6.66s
daCCIB	0.4868	0.4988	0.4722	**0.8441**	0.8584	**0.8220**	0.67s
CondEns[k-means]	0.4961	0.5263	0.4460	0.6557	0.7138	0.5832	0.07s
CondEns[EM]	0.4624	0.5114	0.4255	0.7763	**0.8704**	0.6315	0.09s
CondEns[daIB]	0.4672	0.5202	0.4233	0.7709	0.8696	0.6329	0.82s

Experimental setup

We assume clustering A is known, supplied as categorical known structure, and evaluate the algorithms on their ability to discover clustering B. As the data set is multinomial, we consider performance of CCIB and *CondEns*. The CMBC algorithms assume binary data and so are not evaluated on this set.

In these experiments, 50 synthetic data sets were generated and 10 random initializations of the algorithm were used. We evaluate results in terms of the best, mean, and worst performance over the 10 initializations and average these quantities over the 50 data sets. Results, along with average runtime information, are presented for various settings of K in Tables 8.1 through 8.4.

Analysis of CCIB algorithms

We first examine the results of the CCIB algorithms, focusing on their precision with respect to target partition B. First, and most importantly, we note that the algorithms obtain solutions more similar

Table 8.3: Results for 2d Synthetic Sets: 50 randomly generated data sets with K=4, N=800, M=1100.

$K = 4$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.4532	0.5024	0.4004	0.5868	0.6951	0.4726	27.20s
daCCIB	0.4518	0.4649	0.4383	0.6453	0.6771	**0.5867**	1.43s
CondEns[k-means]	0.3815	0.4120	0.3543	0.5634	0.5978	0.5097	0.47s
CondEns[EM]	0.3489	0.3783	0.3212	**0.6797**	**0.7671**	0.5678	0.27s
CondEns[daIB]	0.3788	0.4289	0.3323	0.5696	0.7000	0.4358	1.88s

Table 8.4: Results for 2d Synthetic Sets: 50 randomly generated data sets with K=5, N=1250, M=1600.

$K = 5$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.4103	0.4450	0.3703	0.5282	0.6324	0.4357	69.79s
daCCIB	0.4105	0.4286	0.3910	0.5681	0.6274	**0.5273**	2.57s
CondEns[k-means]	0.3090	0.3274	0.2925	0.4782	0.5076	0.4348	1.00s
CondEns[EM]	0.2898	0.3129	0.2682	**0.5985**	**0.6643**	0.5060	1.02s
CondEns[daIB]	0.3012	0.3370	0.2710	0.4312	0.5022	0.3548	4.17s

to target partitioning B than to known partitioning A, that is, they are finding solutions which are not redundant with respect to the known structure. The algorithms perform well on the simpler sets with low K. For $K = 2$, both CCIB algorithms find the optimal solution, precision=1.000, for at least one initialization. The sequential clustering approach, seqCCIB, finds the optimal solution for all initializations over all data sets. As K increases, performance decreases. At $K = 5$, the mean precision is 0.5282 for seqCCIB and 0.5681 for daCCIB.

Beyond considering the success of the algorithms at discovering the target clustering, one may also examine how similar solutions are to the known partitioning. Ideally, the algorithms should discover solutions which have low precision with respect to known partitioning A. As the number of classes K varies, the baseline for precision also varies. The baseline of a random assignment should be approximately $1/K$. As can be seen in the data, while this is nearly achieved for $K = 2$, it is not attained for the other $K = 3, 4, 5$.

Contrasting the two algorithms, for the simple set where $K = 2$, the sequential clustering version, seqCCIB, outperforms the deterministic annealing version, daCCIB, on average. However, for $K =$

$3, K = 4$ and $K = 5$, the deterministic annealing version consistently has better mean performance, outperforming seqCCIB by 28.8%, 10.0%, and 7.6% respectively.

If one examines the max and min performance of each algorithm, a crucial difference between the two algorithms becomes apparent: the deterministic annealing version is considerably less sensitive to initialization than the sequential clustering version. Table 8.5 shows the average range of precisions obtained over initializations for seqCCIB versus daCCIB. As shown in the table, except for the case where $K = 2$, the range of solutions obtained by seqCCIB is at least 2x as great as that obtained by daCCIB. The relative insensitivity to initialization is a well-known phenomenon attributed to deterministic annealing approaches which we will see replicated throughout our experiments.

Table 8.5: The seqCCIB algorithm is more sensitive to initialization than daCCIB: average range of precision scores for 10 initializations on the 2d synthetic sets.

	max - min $Prec_B(C)$	
K	seqCCIB	daCCIB
$K = 2$	0.0000	0.4604
$K = 3$	0.3605	0.0364
$K = 4$	0.2225	0.0904
$K = 5$	0.1967	0.1001

A further important difference between the two approaches is the runtime. On the simple set with $K = 2$, seqCCIB is substantially faster than daCCIB, however, for larger sets with more complicated structure, the complexity advantages of daCCIB become apparent. For $K = 3, 4, 5$ the seqCCIB algorithm is 10x, 19x, and 27x slower than daCCIB respectively. This is another phenomenon which we will find consistent throughout the experimental section

Our conclusions from this round of experiments with daCCIB and seqCCIB are that on non-trivial sets, daCCIB consistently shows better mean performance, is more stable with respect to initialization, and has considerably better runtimes which are an order of magnitude better than seqCCIB.

Analysis of *CondEns* algorithms

We now examine performance of the *CondEns* approaches. As with CCIB, performance decreases for higher K. At $K = 2$, *CondEns*, using EM and IB base clustering methods, achieve mean performance better than 0.91, however by $K = 5$, the EM version is obtaining a precision score of 0.60 and IB is obtaining only 0.43.

Comparing the various base-clustering techniques, it is immediately apparent that k-means substantially underperforms the EM and IB techniques for low K. At $K = 2$, k-means scores only 0.7643 whereas EM and IB achieve 0.9145 and 0.9173. Furthermore, the solutions obtained by k-means are quite similar to the known categorization A, with mean $Prec_A(C) = 1.7403$. As K increases, the performance using k-means improves somewhat, outdoing the performance of IB, however it never rivals the performance of EM. For $K = 5$, k-means scores a mean $Prec_B(C)$ of 0.4782 to the IB score of 0.4312, but EM exceeds both with 0.5985.

The sensitivity to initialization may be examined in Table 8.6. The range of precisions $Prec_B(C)$ obtained by *CondEns[EM]* and *CondEns[daIB]* is larger than that of *CondEns[k-means]* by approximately a factor of 2. While *CondEns[k-means]* shows less sensitivity to initialization, this comes at a cost as the precisions obtained are substantially lower than *CondEns[EM]*. For the *CondEns*

Table 8.6: Sensitivity to Initialization

	max - min $Prec_B(C)$		
K	*CondEns[k-means]*	*CondEns[EM]*	*CondEns[daIB]*
$K = 2$	0.0426	0.1098	0.1127
$K = 3$	0.1306	0.2389	0.2367
$K = 4$	0.0881	0.1993	0.2642
$K = 5$	0.0728	0.1583	0.1474

techniques, the runtimes are of similar magnitudes. For $K = 2, 3, 4$, and 5, the *CondEns[daIB]* times take on average 12.5x, 9.1x, 7.0x, and 4.1x more time than *CondEns[EM]* which shows that as K increases, the relative computational advantage of *CondEns[EM]* versus *CondEns[daIB]* weakens. It is not surprising that *CondEns[daIB]* would require longer runtime as the base clustering method, daIB, is a deterministic annealing technique the base clustering method EM, used in *CondEns[EM]*

does not use deterministic annealing and so should be considerably faster.

For the *CondEns* family, we conclude that *CondEns[EM]* is the best performer of the three base clustering techniques considered, while also boasting runtimes which are comparable to *CondEns[k-means]* and several times faster than *CondEns[daIB]*.

Analysis of CCIB versus *CondEns*

We now compare the CCIB algorithms to the *CondEns* family with a particular focus on daCCIB and *CondEns[EM]* which, in general, show the best mean precision and runtimes of their respective families. Looking at mean precision, $Prec_B(C)$, *CondEns[EM]* slightly outscores daCCIB for $K = 2, 4$, and 5. Specifically, it outscores daCCIB by 8.0%, 5.3% and 5.4% respectively. For $K = 3$, daCCIB outscores *CondEns[EM]* by 8.7%.

Another difference emerges if the $Prec_A(C)$ scores are considered. These scores measure the similarity to the known, undesired, solution A and are reviewed in Table 8.7 along with the ratio of the two. It is interesting to observe that as K increases, *CondEns[EM]* actually finds solutions which are *less like* B than does daCCIB. By $K = 5$, the daCCIB solutions have average precisions with respect to the known categorization that are 41.65% higher than *CondEns[EM]*.

Table 8.7: Comparing mean precision, $Prec_A(C)$, to known categorization (*not the target clustering,* B) for daCCIB and *CondEns[EM]*.

	mean $Prec_A(C)$		
K	daCCIB	*CondEns[EM]*	daCCIB/*CondEns[EM]*
$K = 2$	0.5424	0.5933	0.9142
$K = 3$	0.4868	0.4624	1.0528
$K = 4$	0.4518	0.3489	1.2949
$K = 5$	0.4105	0.2898	1.4165

Another important distinction between the algorithms is the range of solutions obtained. This range is greater for *CondEns[EM]* than daCCIB. The ranges are reviewed in Table 8.8. This greater range manifests itself in the higher max $Prec_B(C)$ scores of *CondEns[EM]*, which outscores daCCIB for $K = 3, 4$, and 5. It also is responsible for daCCIB having higher min $Prec_B(C)$ scores on the same values for K.

Table 8.8: Sensitivity to Initialization

	max - min $Prec_B(C)$	
K	daCCIB	*CondEns[EM]*
$K = 2$	0.4604	0.1098
$K = 3$	0.0364	0.2389
$K = 4$	0.0904	0.1993
$K = 5$	0.1001	0.1583

Finally, examining the runtimes shows that *CondEns[EM]* is consistently faster than daCCIB for all K considered. The average runtime for daCCIB is 62.5x, 7.4x, 5.3x, and 2.5x longer than *CondEns[EM]* for $K = 1, 2, 3, 4$ respectively.

In conclusion, we observe that the mean performance of daCCIB and *CondEns[EM]* are competitive, with daCCIB outperforming *CondEns[EM]* on low K and *CondEns[EM]* outperforming daCCIB on high K. The range of solutions obtained by *CondEns[EM]*, however is consistently greater than that of daCCIB. Whether this is an advantage or disadvantage depends on the application being considered. On the one hand, it allows *CondEns[EM]* to obtain some solutions which are better than all of those obtained by daCCIB, as evidenced by the max $Prec_B(C)$ scores. On the other hand, it also means that *CondEns[EM]* will obtain solutions worse than all of those obtained by daCCIB, as evidenced by the min $Prec_B(C)$ scores. Another important consideration to note is the $Prec_A(C)$ scores which favor *CondEns[EM]* for the data sets with higher K. Finally, we find a consistent advantage in runtime for *CondEns[EM]* versus daCCIB.

8.3.2 Higher dimensional synthetic sets

Binary synthetic set generation

We generate test data sets for N instances with binary features $y \in \{0, 1\}^m$. Two independent partitionings A and B with K clusters are embedded in the data by associating A with m_A of the features and B with m_B of the features, making A the dominant partitioning. Noise is introduced by randomly flipping each binary feature with probability $p_{noise} = 0.1$. For specific details, see Appendix E.1. An example set is in Table 8.9. The resulting sets should contain natural partitionings

A and B where the relative strength of the two partitionings can be altered by varying m_A and m_B. The sets are binary and so are appropriate for CMBC, CCIB, and *CondEns*. We evaluate the algorithms on data sets where the relative strength of A and B is varied, as the number of clusters K is varied, and as the amount of noise from p_{noise} is varied.

Table 8.9: Sample binary synthetic set: $N = 8, M_A = 6, M_B = 4, p_{noise} = 0.1$. Results of noise are in bold.

	A						B			
x_1	0	0	0	**0**	1	1	0	0	1	1
x_2	0	0	0	1	1	1	**1**	0	1	1
x_3	1	1	1	0	0	0	0	0	1	1
x_4	1	1	1	**1**	0	0	1	1	0	0
x_5	1	1	1	0	0	0	1	1	0	0
x_6	0	0	0	1	1	1	0	0	1	1
x_7	1	1	**0**	**1**	0	0	0	0	1	1
x_8	0	0	0	1	1	1	**1**	0	1	1

Experimental design

The binary synthetic sets are used in order to provide a controlled setting in which to explore the performance of all algorithms as parameters of the data set are varied. First, the relative strength of partitions A and B will be varied in order to evaluate how weak the secondary structure B may be with respect to A and still be discovered by the algorithms. Next, we will examine the impact of noise within the data set. Finally, we will consider the performance as the number of clusters, K, is varied.

In order to test CMBC, the parameter γ, which controls the tradeoff between the likelihood and the penalty term, must be set. We determine γ by optimizing it over possible settings for a base setting of synthetic data generation parameters. That γ is then fixed and used for all generation parameters. Optimizing for γ requires that the solution be known in advance for the base parameter setting. Despite the fact this is typically not the case in practice, we use this approach as it allows us to compare the algorithms' optimal performance. In later sections, we will consider the parameter dependence of CMBC.

Table 8.10: Results for binary synthetic sets: 10 randomly generated data sets with $K = 2, N = 200, m_A = 4, m_B = 4$. Algorithms are run with 10 random initializations for each data set. Results are shown for the mean, max and min precision over initializations, averaged over all 10 sets. Highest scores are in bold.

$K = 2, dims = [4, 4]$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.5260	0.5585	0.5150	0.8311	0.9765	0.5285	2.67s
daCCIB	0.5270	0.5270	0.5270	**0.9755**	0.9755	**0.9755**	0.27s
CondEns[k-means]	0.5224	0.5505	0.5115	0.9685	**0.9800**	0.9330	0.02s
CondEns[EM]	0.5227	0.5330	0.5135	0.9591	0.9755	0.8720	0.04s
CondEns[daIB]	0.5226	0.5295	0.5125	0.9716	0.9760	0.9660	0.32s
CMBC-bu	0.5182	0.5535	0.5070	0.9478	0.9740	0.8090	0.08s
CMBC-pm	0.5161	0.5310	0.5095	0.9645	0.9740	0.9040	0.15s
daCMBC-bu	0.5148	0.5150	0.5145	0.9707	0.9710	0.9705	0.89s
daCMBC-pm	0.5150	0.5150	0.5145	0.9706	0.9710	0.9705	1.98s

Table 8.11: Results for binary synthetic sets: 10 randomly generated data sets with $K = 2, N = 200, m_A = 10, m_B = 4$.

$K = 2, dims = [10, 4]$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.5734	0.7440	0.5240	0.8527	0.9770	0.5765	3.51s
daCCIB	0.5335	0.5335	0.5335	0.9750	0.9750	0.9750	0.33s
CondEns[k-means]	0.5323	0.5405	0.5260	0.9752	0.9820	0.9690	0.02s
CondEns[EM]	0.5371	0.5555	0.5195	0.9379	0.9785	0.7770	0.04s
CondEns[daIB]	0.5338	0.5400	0.5235	0.9698	0.9785	0.9565	0.34s
CMBC-bu	0.5454	0.6510	0.5185	0.9169	**0.9855**	0.6700	0.13s
CMBC-pm	0.5255	0.5315	0.5205	0.9785	**0.9855**	0.9635	0.28s
daCMBC-bu	0.5256	0.5270	0.5255	**0.9791**	0.9805	**0.9790**	1.49s
daCMBC-pm	0.5255	0.5255	0.5255	0.9790	0.9790	**0.9790**	3.38s

Table 8.12: Results for binary synthetic sets: 10 randomly generated data sets with $K = 2, N = 200, m_A = 30, m_B = 4$.

$K = 2, dims = [30, 4]$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.9252	0.9905	0.6145	0.5711	0.8470	0.5195	3.60s
daCCIB	0.9942	0.9960	0.9935	0.5306	0.5325	0.5295	0.23s
CondEns[k-means]	0.5218	0.5330	0.5130	0.9691	0.9775	0.9615	0.02s
CondEns[EM]	0.5327	0.5610	0.5110	0.7477	0.9430	0.5270	0.04s
CondEns[daIB]	0.5234	0.5500	0.5110	0.9456	0.9670	0.9080	0.45s
CMBC-bu	0.5423	0.6510	0.5110	0.9069	0.9795	0.6385	0.32s
CMBC-pm	0.5222	0.5450	0.5125	0.9658	**0.9800**	0.8995	0.69s
daCMBC-bu	0.5208	0.5235	0.5195	0.9722	0.9740	0.9695	2.87s
daCMBC-pm	0.5206	0.5225	0.5200	**0.9728**	0.9740	**0.9715**	6.61s

Table 8.13: Results for binary synthetic sets: 10 randomly generated data sets with $K = 2, N = 200, m_A = 100, m_B = 4$.

$K = 2, dims = [100, 4]$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.9995	0.9995	0.9995	0.5365	0.5365	0.5365	3.47s
daCCIB	1.0000	1.0000	1.0000	0.5370	0.5370	0.5370	0.33s
CondEns[k-means]	0.5500	0.6155	0.5260	0.9347	**0.9745**	0.7405	0.04s
CondEns[EM]	0.5340	0.5340	0.5340	0.5360	0.5360	0.5360	0.04s
CondEns[daIB]	0.5832	0.6340	0.5385	0.8454	0.9180	0.7405	0.74s
CMBC-bu	0.6280	0.7235	0.5510	0.5889	0.7835	0.5135	0.41s
CMBC-pm	0.5976	0.6825	0.5380	0.6421	0.8480	0.5195	1.04s
daCMBC-bu	0.5336	0.5370	0.5310	**0.9663**	0.9700	**0.9610**	3.90s
daCMBC-pm	0.5765	0.8150	0.5315	0.9283	0.9725	0.7060	9.37s

Results for $K = 2$, $K = 3$, and $K = 4$ are given in Tables 8.10 through 8.21. We restrict K to this range as on experiments with large K, CMBC frequently converges to degenerate solutions with empty clusters. This is a liability of CMBC. Despite this liability, the performance of CMBC with lower K is impressive and so should be examined. We will first consider the results individually for each family of algorithms and then compare across each algorithmic family.

Analysis of CCIB algorithms

We first examine the results of the CCIB family of algorithms by looking at the precision with respect to target partition B for the experiments with $K = 2$ pictured in Tables 8.10 through 8.13 which vary the strength of the dominant partition A by setting the number of features associated with A, m_A. Figure 8.2(a) shows the mean performance of the CIB algorithms for each value of m_A.

First we note that for $m_A \leq 20$, where there are 5 times as many features associated with the known partition as the target partition, the CCIB algorithms still successfully obtain solutions more similar to the target partition. For $m_A = 4$ the mean $Prec_B(C)$ is 0.8311 for seqCCIB and 0.9755 for daCCIB. Compare this to the mean $Prec_A(C)$, which is 0.5734 for seqCCIB and 0.5335 for daCCIB. This is the desired behavior, where both algorithms are obtaining solutions similar to B and unlike A. For $m_A = 10$, the mean $Prec_B(C)$ is 0.8527 for seqCCIB and 0.9750 for daCCIB. Compare this to the mean $Prec_A(C)$, which is 0.5734 for seqCCIB and 0.5335 for daCCIB. This is the desired

Table 8.14: Results for binary synthetic sets: 10 randomly generated data sets with $K = 3, N = 450, m_A = 4, m_B = 4$. Algorithms are run with 10 random initializations for each data set. Results are shown for the mean, max and min precision over initializations, averaged over all 10 sets. Highest scores are in bold.

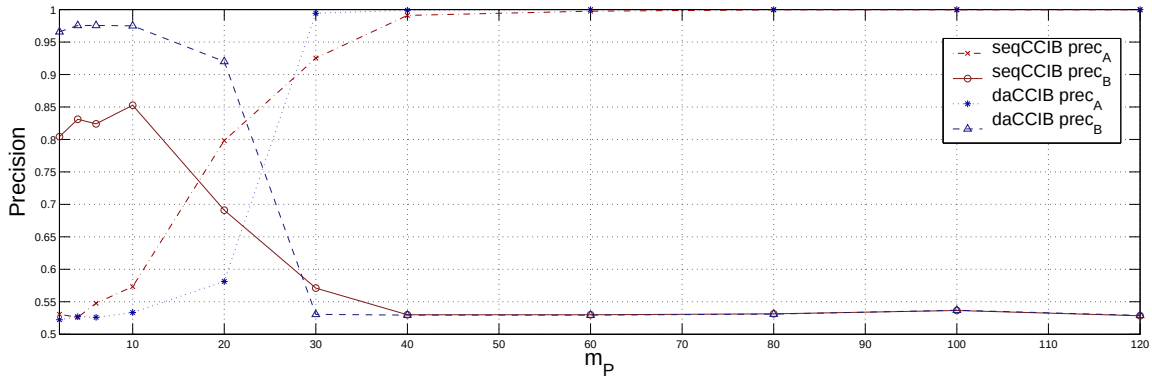
$K = 3, dims = [4, 4]$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.4115	0.4791	0.3593	0.6390	0.8542	0.4753	10.31s
daCCIB	0.3662	0.3784	0.3551	0.8470	0.9078	0.7127	0.50s
CondEns[k-means]	0.3872	0.4900	0.3553	0.8249	**0.9167**	0.6056	0.09s
CondEns[EM]	0.3704	0.3927	0.3600	0.6338	0.7349	0.5813	0.22s
CondEns[daIB]	0.3766	0.4351	0.3573	0.8388	0.9049	0.6273	0.70s
CMBC-bu	0.3694	0.4071	0.3456	0.7381	0.8978	0.5749	0.07s
CMBC-pm	0.3637	0.3889	0.3478	0.8042	0.9131	0.5902	0.27s
daCMBC-bu	0.3612	0.3696	0.3551	0.8113	0.9091	0.6382	1.54s
daCMBC-pm	0.3636	0.3698	0.3556	**0.8814**	0.9144	**0.7933**	4.24s

Table 8.15: Results for binary synthetic sets: 10 randomly generated data sets with $K = 3, N = 450, m_A = 10, m_B = 4$.

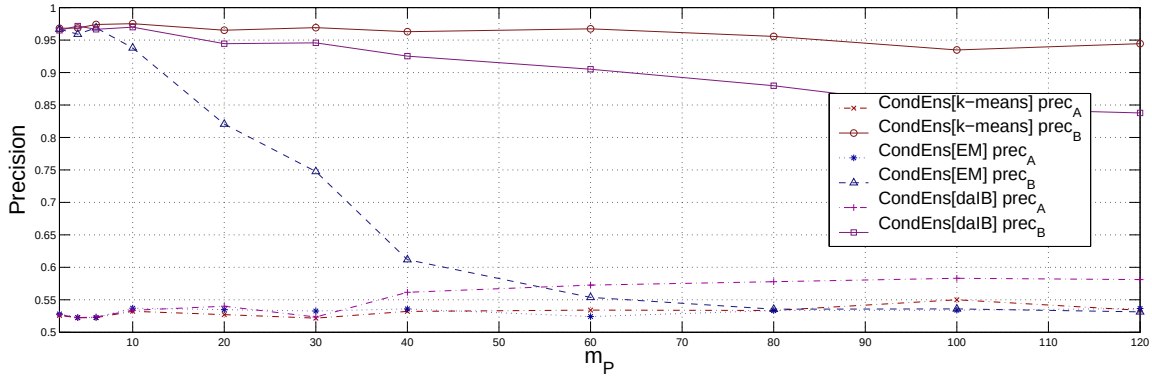
$K = 3, dims = [10, 4]$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.4532	0.5418	0.3718	0.6354	0.8149	0.4698	15.44s
daCCIB	0.3749	0.3960	0.3600	0.8471	0.8916	0.7284	0.67s
CondEns[k-means]	0.3771	0.4522	0.3560	0.8460	**0.9160**	0.6331	0.10s
CondEns[EM]	0.3786	0.4178	0.3540	0.5743	0.6493	0.4329	0.24s
CondEns[daIB]	0.3768	0.4189	0.3578	0.8376	0.8976	0.6436	0.88s
CMBC-bu	0.3826	0.4280	0.3518	0.6370	0.8276	0.5231	0.10s
CMBC-pm	0.3674	0.4042	0.3489	0.7223	0.8878	0.5502	0.32s
daCMBC-bu	0.3612	0.3722	0.3527	0.8395	0.9060	0.6716	2.82s
daCMBC-pm	0.3611	0.3680	0.3556	**0.8915**	0.9122	**0.8291**	8.11s

Table 8.16: Results for binary synthetic sets: 10 randomly generated data sets with $K = 3, N = 450, m_A = 30, m_B = 4$.

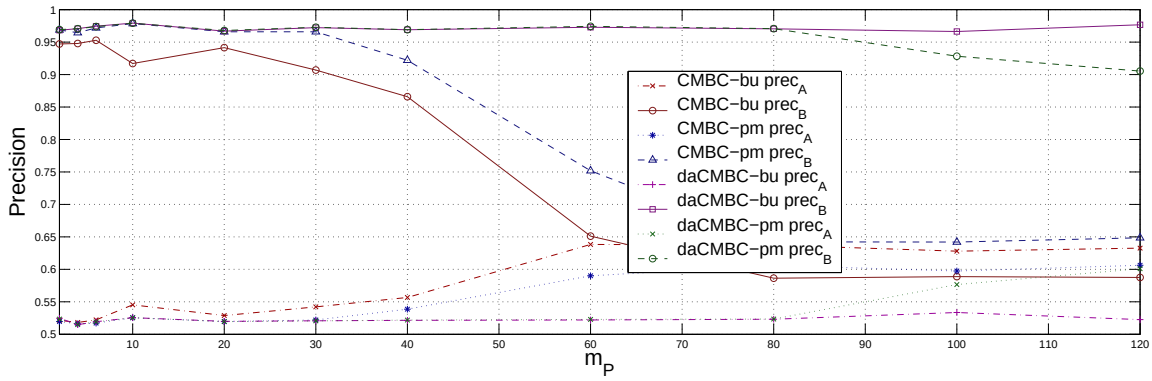
$K = 3, dims = [30, 4]$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.9292	0.9764	0.7578	0.3910	0.5004	0.3576	18.99s
daCCIB	0.8940	0.9071	0.8653	0.4160	0.4362	0.4047	0.52s
CondEns[k-means]	0.3757	0.4307	0.3556	**0.8531**	**0.9147**	0.6491	0.16s
CondEns[EM]	0.3564	0.3593	0.3553	0.3790	0.4416	0.3627	0.09s
CondEns[daIB]	0.4443	0.5247	0.3931	0.6909	0.8087	0.5300	1.48s
CMBC-bu	0.4075	0.4798	0.3609	0.5424	0.6838	0.3900	0.24s
CMBC-pm	0.3809	0.4404	0.3487	0.6451	0.8187	0.4687	0.77s
daCMBC-bu	0.3630	0.3731	0.3513	0.7991	0.9100	0.6118	5.52s
daCMBC-pm	0.3638	0.3718	0.3562	0.8505	0.9127	**0.6758**	16.24s



(a) CCIB algorithms



(b) *CondEns* algorithms



(c) CMBC algorithms

Figure 8.2: Mean precision scores for $K = 2$ where the strength of partition A, m_A, varies.

Table 8.17: Results for binary synthetic sets: 10 randomly generated data sets with $K = 3, N = 450, m_A = 100, m_B = 4$.

$K = 3, dims = [100, 4]$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.9867	0.9867	0.9867	0.3687	0.3733	0.3647	17.16s
daCCIB	0.9827	0.9993	0.8682	0.3693	0.3758	0.3667	0.77s
CondEns[k-means]	0.4118	0.5242	0.3631	**0.7734**	0.8942	0.5644	0.18s
CondEns[EM]	0.3538	0.3538	0.3538	0.3553	0.3553	0.3553	0.09s
CondEns[daIB]	0.5388	0.6718	0.4318	0.4463	0.5260	0.3829	2.27s
CMBC-bu	0.4464	0.5362	0.3771	0.4070	0.5031	0.3589	0.64s
CMBC-pm	0.4298	0.5333	0.3640	0.4457	0.5893	0.3618	2.07s
daCMBC-bu	0.3677	0.3804	0.3533	0.7732	**0.9136**	**0.5802**	6.73s
daCMBC-pm	0.4940	0.9100	0.3569	0.7134	0.9044	0.4118	19.97s

behavior, where again both algorithms are obtaining solutions unlike the known categorization A. We note that performance remains steady for $m_A \leq 10$ and does not begin to drop off until reaching $m_A = 20$.

Still considering the regime of $m_A \leq 20$, we note that again that in terms of $Prec_B(C)$ the daCCIB algorithm outperforms seqCCIB on average. Again, the range of precisions obtained over the initializations varies considerably more for seqCCIB than for daCCIB. At $m_A = 4$, the range is 0.4480 for seqCCIB whereas it is 0 for daCCIB. At $m_A = 10$, the range is 0.4005 for seqCCIB whereas it is again 0 for daCCIB. For both of these settings, the maximum seqCCIB score only slightly ahead of the mean daCCIB score, with a gain of 0.1% for $m_A = 4$ and 0.21% for $m_A = 10$. This confirms the general trend seen in comparing the two, where daCCIB boasts better mean performance and much less variation in solutions than seqCCIB.

Now considering $m_A \geq 30$, it is apparent that performance degrades substantially. For these settings the algorithms find solutions which are more similar to known categorization A than to B. At $m_A = 30$, mean $Prec_B(C)$ is only 0.5711 for seqCCIB and 0.5306 for daCCIB whereas mean $Prec_A(C)$ is considerably higher at 0.9252 for seqCCIB and 0.9942 for daCCIB. This is in part a consequence of the coordination term which rewards high $\lambda I(C;Y)$. A solution $C = A$ may in fact optimize the CCIB objective function. Consider two possible solutions, $C = A$ and $C = B$. For these data sets as m_A grows, at a certain point, the contribution from $I(A;Y|A) + \lambda I(A;Y) = \lambda I(A;Y)$

Table 8.18: Results for binary synthetic sets: 10 randomly generated data sets with $K = 4, N = 800, m_A = 4, m_B = 4$. Algorithms are run with 10 random initializations for each data set. Results are shown for the mean, max and min precision over initializations, averaged over all 10 sets. Highest scores are in bold.

$K = 4, dims = [4, 4]$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.3465	0.3889	0.3084	0.4763	0.5946	0.3872	26.08s
daCCIB	0.2908	0.3261	0.2733	0.7302	0.7941	0.6199	0.48s
CondEns[k-means]	0.3039	0.3825	0.2732	0.6985	0.7977	0.5619	0.47s
CondEns[EM]	0.2847	0.2964	0.2789	0.2877	0.3553	0.2767	0.63s
CondEns[daIB]	0.2939	0.3392	0.2732	0.7127	**0.8034**	0.5607	1.01s
CMBC-bu	0.2938	0.3315	0.2687	0.6154	0.7556	0.4755	0.09s
CMBC-pm	0.2879	0.3190	0.2701	0.6337	0.7696	0.5020	0.29s
daCMBC-bu	0.2784	0.2883	0.2684	0.7162	0.7981	0.5915	1.68s
daCMBC-pm	0.2802	0.2910	0.2693	**0.7736**	0.8026	**0.6807**	5.98s

Table 8.19: Results for binary synthetic sets: 10 randomly generated data sets with $K = 4, N = 800, m_A = 10, m_B = 4$.

$K = 4, dims = [10, 4]$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.3625	0.4404	0.3155	0.4782	0.6310	0.3785	45.95s
daCCIB	0.3097	0.3330	0.2810	0.6994	0.7745	0.6241	0.87s
CondEns[k-means]	0.2995	0.3580	0.2731	0.7209	0.7986	0.5624	0.43s
CondEns[EM]	0.2816	0.3111	0.2745	0.2725	0.2826	0.2686	0.47s
CondEns[daIB]	0.3121	0.3828	0.2775	0.6858	0.7900	0.5651	1.76s
CMBC-bu	0.3088	0.3675	0.2746	0.5056	0.6333	0.3961	0.13s
CMBC-pm	0.2981	0.3379	0.2718	0.5491	0.6687	0.4411	0.47s
daCMBC-bu	0.2764	0.2863	0.2686	0.7651	0.8001	**0.6651**	4.17s
daCMBC-pm	0.2767	0.2853	0.2698	**0.7719**	**0.8023**	0.6637	15.49s

Table 8.20: Results for binary synthetic sets: 10 randomly generated data sets with $K = 4, N = 800, m_A = 30, m_B = 4$.

$K = 4, dims = [30, 4]$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.9135	0.9461	0.7648	0.2883	0.3537	0.2705	69.74s
daCCIB	0.9072	0.9377	0.8194	0.2908	0.3245	0.2756	1.35s
CondEns[k-means]	0.3030	0.3494	0.2781	0.7121	0.7991	0.5709	0.58s
CondEns[EM]	0.2717	0.2718	0.2716	0.2674	0.2674	0.2674	0.21s
CondEns[daIB]	0.5073	0.6461	0.4090	0.4293	0.5469	0.3435	3.51s
CMBC-bu	0.3357	0.3965	0.2894	0.3818	0.4596	0.3173	0.33s
CMBC-pm	0.3267	0.3842	0.2891	0.4109	0.5011	0.3346	1.24s
daCMBC-bu	0.2791	0.2870	0.2699	0.7553	**0.8027**	0.6105	8.76s
daCMBC-pm	0.2792	0.2880	0.2707	**0.7801**	0.8025	**0.7010**	34.10s

Table 8.21: Results for binary synthetic sets: 10 randomly generated data sets with $K = 4, N = 800, m_A = 100, m_B = 4$.

$K = 4, dims = [100, 4]$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.9710	0.9850	0.8812	0.2777	0.2836	0.2730	57.58s
daCCIB	0.9438	0.9948	0.8063	0.2802	0.2865	0.2754	1.69s
CondEns[k-means]	0.3762	0.4884	0.3003	0.6134	0.7684	0.4329	0.96s
CondEns[EM]	0.2737	0.2737	0.2737	0.2669	0.2669	0.2669	0.25s
CondEns[daIB]	0.5642	0.6947	0.4364	0.2999	0.3254	0.2779	4.73s
CMBC-bu	0.3670	0.4751	0.2899	0.2955	0.3372	0.2713	0.93s
CMBC-pm	0.3603	0.4691	0.2891	0.2923	0.3271	0.2704	4.34s
daCMBC-bu	0.3215	0.3634	0.2876	**0.7197**	**0.7796**	**0.5571**	10.64s
daCMBC-pm	0.7801	0.9874	0.3615	0.4053	0.7004	0.2751	46.20s

will eventually outweigh the contribution from $I(B;Y|A) + \lambda I(B;Y)$. This shows that for solutions where the dominant clustering is an order of magnitude stronger than the secondary clustering, the solutions obtained may be sensitive to the λ chosen.

Table 8.22: Ratio of mean run time required by seqCCIB to daCCIB for various K and m_A.

	$m_A = 4$	$m_A = 10$	$m_A = 30$	$m_A = 100$
$K = 2$	9.89x	10.63x	15.65x	10.51x
$K = 3$	20.62x	23.05x	36.52x	22.29x
$K = 4$	54.33x	52.82x	51.66x	34.23x

Finally, examining the runtimes of the algorithms confirms our previous observations. The ratios of runtimes of seqCCIB to daCCIB for various K and m_A are shown in Table 8.22. For $K = 2$ and $m_A = 4$, daCCIB requires 0.27s whereas seqCCIB requires 2.67s meaning seqCCIB requires 9.89x longer time to run on average. As m_A grows, the runtimes also grow, however the ratios remain relatively steady. The more striking divergence in runtimes is due to increasing K. Whereas the ratios range between 9.89 and 15.65 for $K = 2$, they are between 20.62 and 36.52 for $K = 3$ and between 34.23 and 54.33 for $K = 4$. This highlights a further advantage of daCCIB which for moderately small K can obtain speedups of at least 50x compared to seqCCIB.

Analysis of *CondEns* algorithms

We now consider performance of the *CondEns* family of algorithms. The most striking feature of the results is that both *CondEns[k-means]* and *CondEns[daIB]* obtain solutions similar to target partitioning B for the entire range of m_A considered. Specifically, at $m_A = 4$ where *CondEns[k-means]* has a mean $Prec_B(C)$ score of 0.9685 and *CondEns[daIB]* scores 0.9698. These two algorithms continue to score well up to and through $m_A = 100$ where *CondEns[k-means]* obtains a score of 0.9347 and *CondEns[daIB]* obtains a score of 0.8454. The *CondEns[EM]*, however, performs well for $m_A \leq 30$, however substantially drops off by $m_A = 100$ where it obtains a $Prec_B(C)$ of only 0.5360. This is an issue for further study.

Analysis of CMBC algorithms

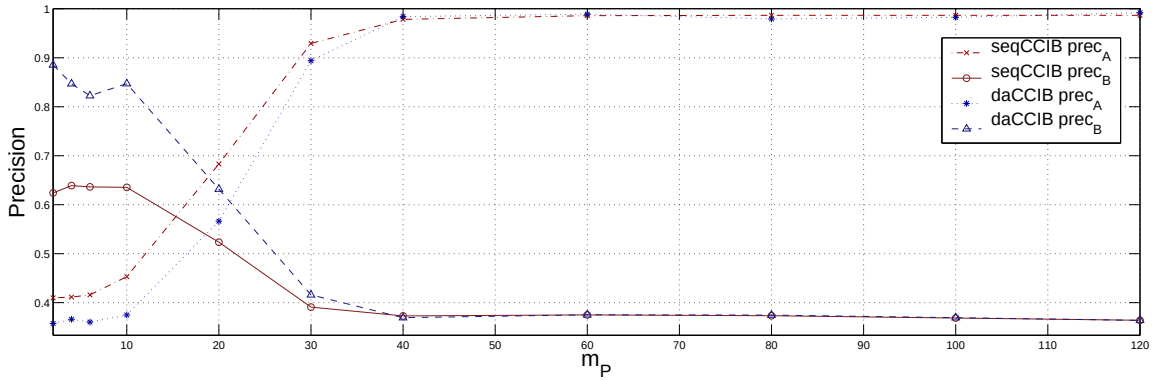
The CMBC algorithms perform well for the entire range of m_B considered. At $m_A = 4$, all four CMBC approaches perform well, with average $Prec_B(C)$ scores ranging from 0.9478 obtained by CMBC-bu to 0.9707 obtained by daCMBC-bu. The CMBC-bu is consistently the lowest scoring of the four for all m_A, however it does perform decently for $m_A = 10$ with 0.9169 and $m_A = 30$ with 0.9069. By $m_A = 100$, however, performance has decreased to 0.5889. The other basic approach, CMBC-pm, shows a similar trend, scoring 0.9785 for $m_A = 10$, 0.9658 for $m_A = 30$, and then dropping off to 0.6421 for $m_A = 100$. The performance of the deterministic annealing algorithms, however, is remarkably resilient to increasing m_B. At $m_A = 10$, daCMBC-bu and daCMBC-pm score 0.9791 and 0.9790 respectively, at $m_A = 30$ they score 0.9722 and 0.9728, and even at $m_A = 100$, they score 0.9663 and 0.9283, showing a decisive advantage over the non-deterministic annealing basic versions. Also notable is the fact that the batch update versions, CMBC-bu and daCMBC-bu, perform competitively with the partial maximization versions, CMBC-pm and daCMBC-pm. While CMBC-bu does consistently underperform CMBC-pm, the daCMBC-bu approach often exceeds the performance of daCMBC-pm. This suggests that the batch update approaches are reasonable approximation with little loss of performance in practice.

Much like the behavior with the CCIB algorithms, the deterministic annealing approaches are substantially less sensitive to initialization than the basic versions. Table 8.23 shows the range of scores obtained by each of the CMBC algorithms for the various m_A and $K = 2$. In general, the deterministic annealing versions have a substantially smaller range of scores. Except for the case $m_A = 100$ where daCMBC-pm has a range of 0.2665, all ranges for daCMBC-bu and daCMBC-pm are less that 0.01. That insensitivity to initialization is in contrast to the basic algorithms where CMBC-bu has ranges that vary from 0.1650 to 0.2700 and CMBC-pm from 0.0220 to 0.3285. For the basic algorithms, there is an apparent difference in the batch update and partial maximization approaches. Clearly the batch update approximation allows for a greater range of solutions to be obtained than the partial maximization approach. This is not the case for the deterministic annealing algorithms, where (ignoring the $M_A = 100$ case) daCMBC-bu is roughly as consistent as daCMBC-pm. Finally, we consider the runtimes of the CMBC algorithms. Not surprisingly,

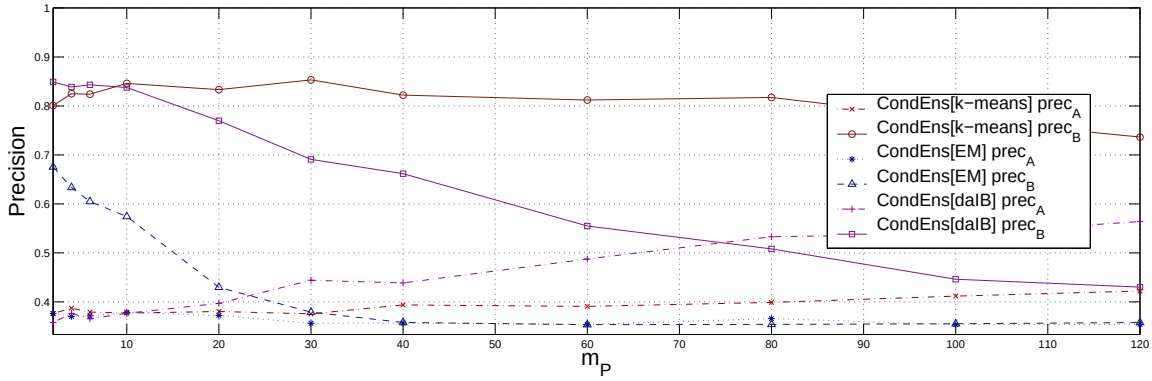
Table 8.23: Sensitivity to Initialization: CMBC algorithms for $K = 2, N = 200, m_A = 4$, and varying m_B.

	max - min $Prec_B(C)$			
K	CMBC-bu	CMBC-pm	daCMBC-bu	daCMBC-pm
$m_A = 4$	0.1650	0.0700	0.0005	0.0005
$m_A = 10$	0.3155	0.0220	0.0015	0.0000
$m_A = 30$	0.3410	0.0805	0.0045	0.0025
$m_A = 100$	0.2700	0.3285	0.0090	0.2665

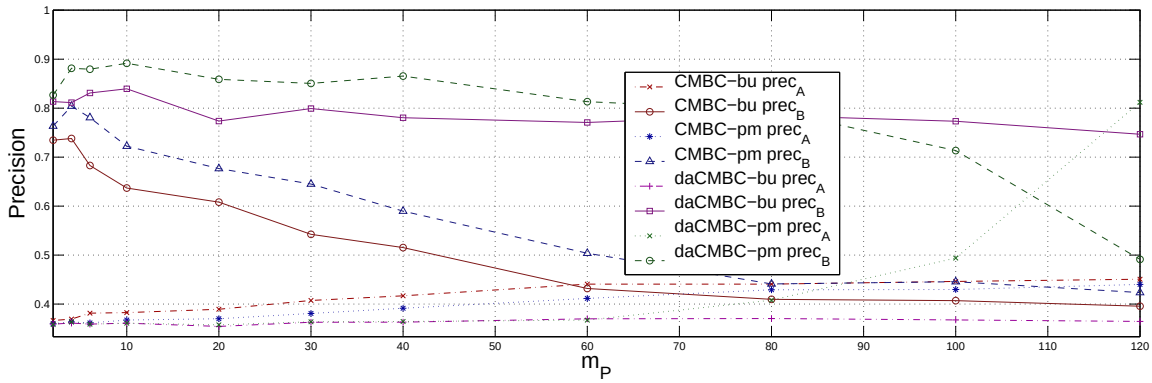
the basic approaches are faster than the deterministic annealing approaches and the batch update versions faster than the partial maximization. In Table 8.24 the ratio of runtimes of the deterministic annealing approaches against the partial maximization version of the basic approach as it obtains competitive quality solutions for lower m_A. The ratio ranges from 5.93x to 3.75x for daCMBC-bu and from 13.20x to 9.01x for daCMBC-pm, in both cases consistently decreasing as the dimension is increased.



(a) CCIB algorithms

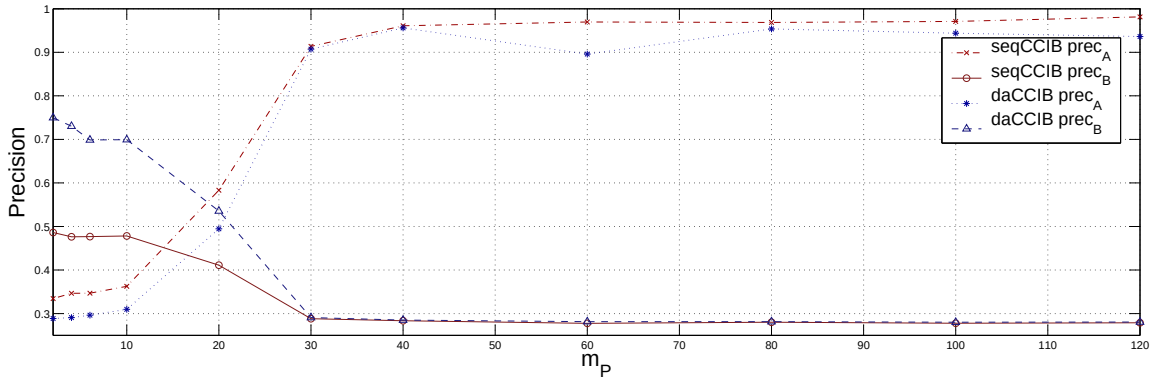


(b) *CondEns* algorithms

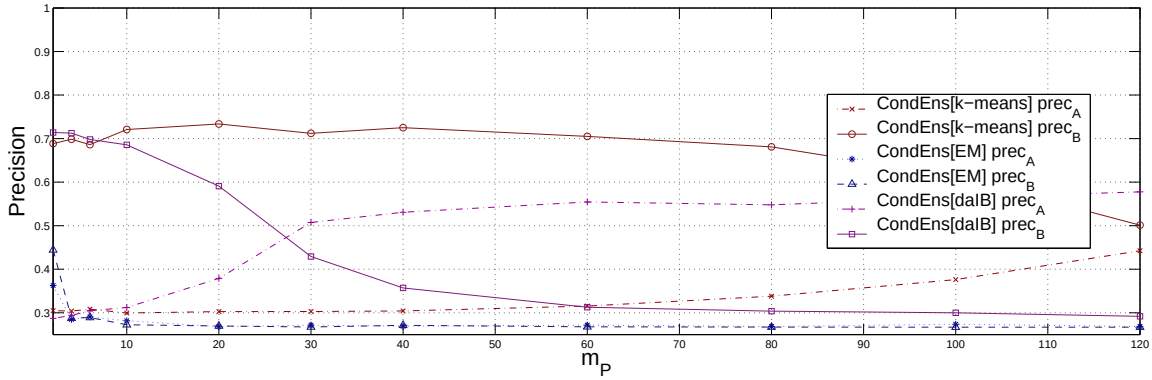


(c) CMBC algorithms

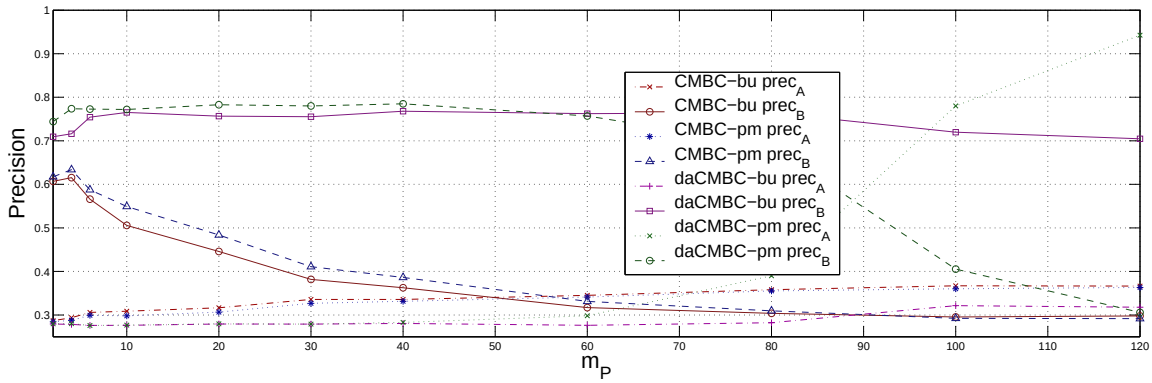
Figure 8.3: Mean precision scores for $K = 3$ where the strength of partition A, m_A, varies.



(a) CCIB algorithms



(b) *CondEns* algorithms



(c) CMBC algorithms

Figure 8.4: Mean precision scores for $K = 4$ where the strength of partition A, m_A, varies.

Table 8.24: Ratio of mean run time required by daCMBC-bu and daCMBC-pm to CMBC-pm for K=2, various m_A.

	$m_A = 4$	$m_A = 10$	$m_A = 30$	$m_A = 100$
daCMBC-bu	5.93x	5.32x	4.16x	3.75x
daCMBC-pm	13.20x	12.07x	9.58x	9.01x

Table 8.25: Confusion matrices for face data.

initial clustering		
	female	male
c_1	140	144
c_2	45	40
Precision = 0.5122		

second clustering		
	female	male
c_1	105	1
c_2	80	183
Precision = 0.7805		

8.4 Real data

8.4.1 Interactive session: face data set

We consider a set of 369 face images with 40×40 grayscale pixels and gender annotations. We performed clustering with $K = 2$ clusters and a Gaussian noise model for the features. Initially, no background knowledge was used. All of 20 trials converged to the same clustering, suggesting that this clustering is the dominant structure in the data set. The precision score between the discovered clustering and the gender classification was 0.5122, i.e. the overlap is close to random. Examining the centroids of each cluster in Figure 8.5 shows the clustering which was obtained partitions the data into face-and-shoulder views and face-only views.

We then introduce the clustering generated from the first attempt as background knowledge and perform a second attempt. The resulting precision score is substantially higher, at 0.7805. Confusion matrices for both clusterings are in Table 8.25. Centroids for this clustering are in Figure 8.6 and confirm that the dominant structure found in the previous attempt has been avoided, revealing lower-order structure that is more informative with respect to gender.

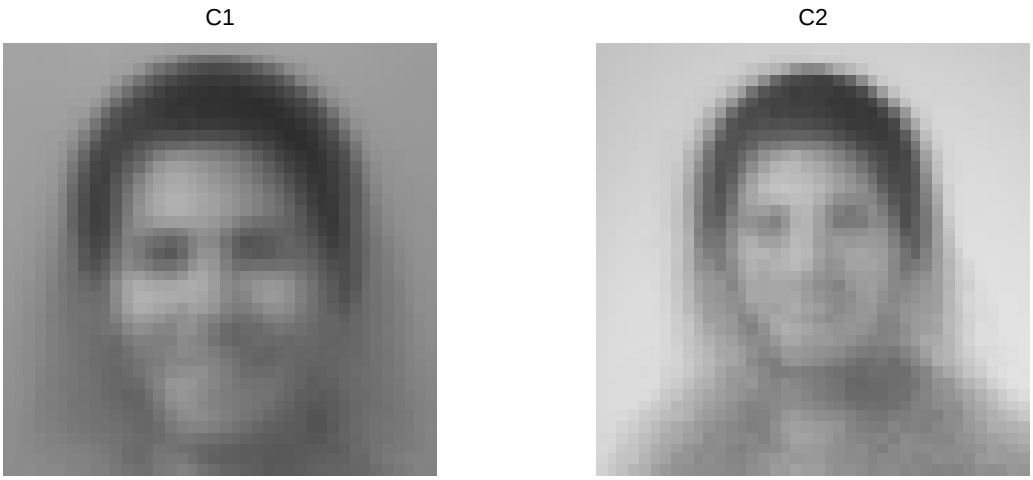


Figure 8.5: Centroids from initial clustering with no side information.

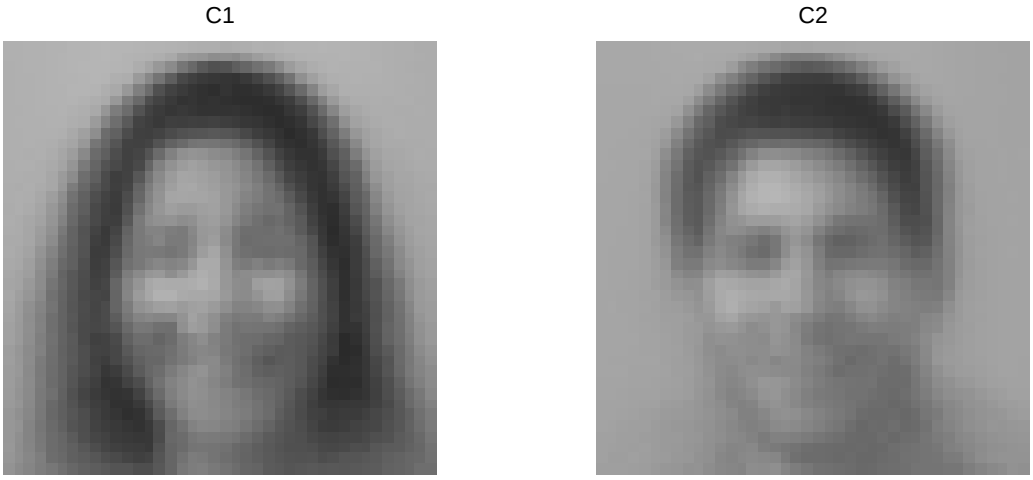


Figure 8.6: Centroids from second clustering using initial clustering as side information.

8.4.2 Text data sets

We evaluate performance on several real-world text data sets. Each may be partitioned according to either one of two independent classification schemes. Experiments are performed using either one of these classification schemes as background knowledge. An algorithm is considered successful if it finds a clustering not associated with the background knowledge, that is similar to the other classification scheme. Documents are represented by term frequency vectors that are assumed to follow a multinomial distribution. For all experiments described, $\lambda = 0.3$ is used and K is set to the cardinality of the target categorization.

The WebKB data set

We use the CMU 4 Universities WebKB data set as described in [Craven et al., 1998] which consists of webpages collected from computer science departments and has a classification scheme based on page type: ('course', 'faculty', 'project', 'staff', 'student') as well source university: ('Cornell', 'Texas', 'Washington', 'Wisconsin'). Documents belonging to the 'misc' and 'other' categories, as well as the 'department' category which contained only 4 members, were removed, leaving 1087 pages remaining. Stopwords were removed, numbers were tokenized, and only terms appearing in more than one document were retained, leaving 5650 terms.

Reuters RCV 1

Additional data sets were derived from the Reuters RCV-1 news corpus which contains multiple labels for each document. We first select a number of topic labels *Topic* and region labels *Region* and then sample documents from the set of documents having labels $\{Topic \times Region\}$. We take up to n documents from each combination of labels. For ease of evaluation, those documents which contain multiple labels from *Topic* or multiple labels from *Region* were excluded. We selected labels which would produce high numbers of eligible documents and generated the following sets:

(i) *RCV1-gmcat2x2: Topic* = { MCAT (Markets), GCAT (Government/Social) } and *Region* = {UK, INDIA}: 1600 documents and 3295 terms.

(ii) *RCV1-ec5x6:* 5 of the most frequent subcategories of the ECAT (Economics) and 6 of the most frequent country codes were chosen: 5362 documents and 4052 terms.

(iii) *RCV1-top7x9:* ECAT (Economics), MCAT (Markets) and the 5 most frequent subcategories of the GCAT (General) topic and the 9 most frequent country codes were chosen: 4345 documents and 4178 terms.

As with the WebKB set, stopwords were removed, numbers were tokenized and a term frequency cutoff was used to remove low-frequency terms.

Experimental Design

We perform tests assuming one of the categorizations is known and evaluate how similar the solution is to the remaining, or *target*, categorization. The number of clusters, K is set to the number of categories in the target categorization. As the number of categories may differ between known and target categorizations, we use Normalized Mutual Information (NMI) which, unlike precision, is well-defined for two clusterings with different K. We evaluate the CCIB and *CondEns* algorithms. CMBC is not evaluated in this session as it often produced degenerate solutions where all instances were assigned to the same cluster. We include results for daCCIB, the deterministic annealing version of CCIB, which is relatively fast. Results for seqCCIB, the sequential clustering version, is not included as it has been shown to typically underperform daCCIB on synthetic sets and takes considerably more time, rendering it impractical for data sets this large. *CondEns[k-means]* and *CondEns[EM]* are considered as they show better performance on synthetic sets and more favorable runtimes with respect to *CondEns[daIB]*. We evaluate each algorithm for 20 random initializations on each data set. Results are shown in Tables 8.26 through 8.33.

Table 8.26: Results on WebKB set with 20 initializations for $Z = L1$ (PageType)

WebKB: $Z = L1$							
Algorithm	$NMI(C, L1)$			$NMI(C, L2)$			runtime
	mean	max	min	mean	max	min	mean
daCCIB	0.0092	0.0126	0.0074	0.0307	0.0864	0.0113	63.46s
CondEns[k-means]	0.0707	0.1057	0.0370	**0.1284**	**0.2535**	0.0167	1.86s
CondEns[EM]	0.0622	0.1131	0.0295	0.0744	0.1435	**0.0400**	1.42s

Table 8.27: Results on WebKB set with 20 initializations for $Z = L2$ (University)

WebKB: $Z = L2$							
Algorithm	$NMI(C, L1)$			$NMI(C, L2)$			runtime
	mean	max	min	mean	max	min	mean
daCCIB	**0.3430**	**0.3885**	**0.2833**	0.0105	0.0224	0.0077	62.85s
CondEns[k-means]	0.2422	0.2892	0.1390	0.0489	0.0852	0.0166	1.80s
CondEns[EM]	0.1689	0.2437	0.1065	0.0284	0.0451	0.0108	1.77s

Table 8.28: Results on RCV1-gmcat2x2 set with 20 initializations for $Z = L1$ (Topic)

RCV1-gmcat2x2: $Z = L1$							
Algorithm	$NMI(C, L1)$			$NMI(C, L2)$			runtime
	mean	max	min	mean	max	min	mean
daCCIB	0.0033	0.0037	0.0020	0.0015	0.0016	0.0005	12.47s
CondEns[k-means]	0.0024	0.0274	0.0000	0.0045	0.0617	0.0002	1.33s
CondEns[EM]	0.0615	0.2506	0.0001	**0.0897**	**0.3076**	**0.0027**	0.50s

Table 8.29: Results on RCV1-gmcat2x2 set with 20 initializations for $Z = L2$ (Region)

RCV1-gmcat2x2: $Z = L2$							
Algorithm	$NMI(C, L1)$			$NMI(C, L2)$			runtime
	mean	max	min	mean	max	min	mean
daCCIB	**0.7891**	**0.7914**	**0.7881**	0.0002	0.0002	0.0001	12.89s
CondEns[k-means]	0.0194	0.1204	0.0019	0.0042	0.0370	0.0000	1.08s
CondEns[EM]	0.4930	0.7656	0.0018	0.0053	0.0437	0.0001	0.45s

Table 8.30: Results on RCV1-ec5x6 set with 20 initializations for $Z = L1$ (Topic)

RCV1-ec5x6: $Z = L1$							
Algorithm	$NMI(C, L1)$			$NMI(C, L2)$			runtime
	mean	max	min	mean	max	min	mean
daCCIB	0.0794	0.0817	0.0753	0.3000	0.3149	**0.2829**	423.34s
CondEns[k-means]	0.1261	0.1491	0.1042	0.2468	0.2837	0.2216	14.65s
CondEns[EM]	0.1503	0.1752	0.1217	**0.3158**	**0.3539**	0.2693	12.69s

Table 8.31: Results on RCV1-ec5x6 set with 20 initializations for $Z = L2$ (Region)

RCV1-ec5x6: $Z = L2$							
Algorithm	$NMI(C, L1)$			$NMI(C, L2)$			runtime
	mean	max	min	mean	max	min	mean
daCCIB	**0.2739**	**0.3122**	**0.2444**	0.2278	0.2430	0.2219	551.67s
CondEns[k-means]	0.1484	0.1997	0.1147	0.2307	0.2368	0.2234	12.23s
CondEns[EM]	0.2526	0.2897	0.2162	0.2273	0.2349	0.2224	12.19s

Table 8.32: Results on RCV1-top6x9 set with 20 initializations for $Z = L1$ (Topic)

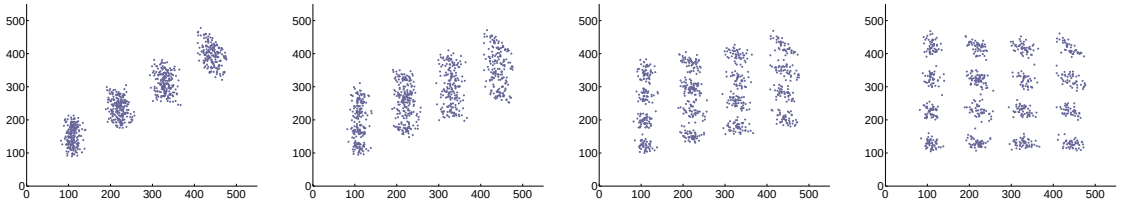
RCV1-top6x9: $Z = L1$							
Algorithm	$NMI(C, L1)$			$NMI(C, L2)$			runtime
	mean	max	min	mean	max	min	mean
daCCIB	0.0268	0.0348	0.0217	**0.1407**	0.1712	**0.1192**	204.83s
CondEns[k-means]	0.2085	0.2641	0.1573	0.1163	**0.1783**	0.0875	19.69s
CondEns[EM]	0.1903	0.2588	0.1332	0.1314	0.1735	0.0964	18.22s

Table 8.33: Results on RCV1-top6x9 set with 20 initializations for $Z = L2$ (Region)

RCV1-top6x9: $Z = L2$							
Algorithm	$NMI(C, L1)$			$NMI(C, L2)$			runtime
	mean	max	min	mean	max	min	mean
daCCIB	0.4388	**0.5483**	0.3805	0.0084	0.0138	0.0050	158.00s
CondEns[k-means]	0.2832	0.3489	0.2215	0.0176	0.0300	0.0149	13.36s
CondEns[EM]	**0.4863**	0.5394	**0.4405**	0.0140	0.0178	0.0083	14.36s

Analysis of Text Data Results

Results of the CCIB and *CondEns* methods on the text data sets are shown in Tables 8.26 through 8.33. It is interesting to note that results improve as sets with more categories are considered. In all cases, the solutions found are more similar to the target classification than the known classification. The performance of the *CondEns* algorithm is competitive with CCIB across the data sets. We will focus on the performance of *CondEns[EM]* which for the most part obtains better solutions than *CondEns[k-means]*. Looking across the data sets, *CondEns[EM]* outscores CCIB in mean performance on half of the sessions. Notably, the range of solutions obtained due to the sensitivity of *CondEns[EM]* to initialization does not appear to help relative to daCCIB. That is, if one considers maximum scores, the relative performance still falls along the lines of mean scores with *CondEns[EM]* obtaining better maximum NMI values again in half of the sessions. A key advantage to *CondEns[EM]*, however is that, daCCIB ranges between 4.57x to 44.67x slower in runtime than *CondEns[EM]*. So while the quality of solutions may not argue for one algorithm over the other, *CondEns[EM]* boasts a substantial advantage in computational efficiency.



(a) $\alpha = .25$ (b) $\alpha = .5$ (c) $\alpha = .75$ (d) $\alpha = 1$

Figure 8.7: Weakening the orthogonality assumption: sample synthetic data sets for $k = 4$ with various α. $n = 800$ items, $m = 1100$ draws

8.5 Robustness to orthogonality assumption

We now evaluate the robustness of the CCIB and *CondEns* algorithms to weakening of the orthogonality assumption. As discussed in the generation procedure given in 8.3.1, orthogonality weight α controls the independence of the two partitions A and B. For $\alpha = 1$, partition membership is independent whereas for $\alpha = 0$ partition membership is completely dependent. Examples of data sets generated with various α are given in Figure 8.7. We evaluate both the CCIB and *CondEns* algorithms as α varies between 0 and 1. Tables 8.34 through 8.45 show the results of the algorithms for $\alpha = 1.0, 0.75, 0.50$, and 0.25 and where $K = 2, 3$, and 4. The mean precisions are plotted in Figures 8.8, 8.9, and 8.10.

Analysis of CCIB algorithms as orthogonality varies.

Directing attention to the CCIB algorithms, the properties of the algorithms seen thusfar are generally confirmed: deterministic annealing shows better mean performance with considerably lower runtimes and less sensitivity to initialization. This has been established for $\alpha = 1$ in Section 8.3.1, so here we briefly review the results for less orthogonal sets (lower α.)

The Tables 8.35, 8.39, and 8.43, which show the results for $\alpha = 0.75$ for $K = 2, 3$, and 4, show mixed results. For $K = 2$ [in Table 8.35], daCCIB outscores seqCCIB with daCCIB's mean precision $Prec_B(C)$ of 0.8509 versus seqCCIB's $Prec_B(C)$ of 0.7761. The seqCCIB algorithm,

Table 8.34: Results for binary synthetic sets: 50 randomly generated data sets with $K = 2, N = 200, M = 400$, $\alpha = 1.00$. Algorithms are run with 10 random initializations for each data set. Results are shown for the mean, max and min precision over initializations, averaged over all 10 sets. Highest scores are in bold.

$K = 2, \alpha = 1.00$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.5431	0.6085	0.5205	0.8799	**1.0000**	0.5805	1.45s
daCCIB	0.5315	0.5315	0.5315	**1.0000**	1.0000	**1.0000**	0.17s
CondEns[k-means]	0.7389	0.7655	0.7220	0.7676	0.7845	0.7410	0.02s
CondEns[EM]	0.5845	0.6355	0.5370	0.9260	0.9765	0.8710	0.02s
CondEns[daIB]	0.5829	0.6440	0.5365	0.9278	0.9770	0.8625	0.25s

Table 8.35: Results: 50 randomly generated data sets with orthogonality weight $\alpha = 0.75$.

$K = 2, \alpha = 0.75$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.6945	0.9810	0.5210	0.7761	**1.0000**	0.5230	1.55s
daCCIB	0.6673	0.6810	0.6165	**0.8509**	0.9080	**0.8365**	0.26s
CondEns[k-means]	0.7507	0.7735	0.7330	0.7558	0.7735	0.7330	0.02s
CondEns[EM]	0.7494	0.7730	0.7330	0.7571	0.7735	0.7335	0.02s
CondEns[daIB]	0.7538	0.7730	0.7360	0.7527	0.7705	0.7335	0.29s

Table 8.36: Results: 50 randomly generated data sets with orthogonality weight $\alpha = 0.50$.

$K = 2, \alpha = 0.50$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.9903	1.0000	0.9030	0.5313	0.5315	0.5295	1.56s
daCCIB	1.0000	1.0000	1.0000	0.5315	0.5315	0.5315	0.19s
CondEns[k-means]	0.7503	0.7735	0.7335	**0.7556**	0.7730	**0.7320**	0.02s
CondEns[EM]	0.7503	0.7735	0.7285	0.7554	**0.7780**	**0.7320**	0.02s
CondEns[daIB]	0.7460	0.7640	0.7085	0.7457	0.7640	0.7050	0.38s

Table 8.37: Results: 50 randomly generated data sets with orthogonality weight $\alpha = 0.25$.

$K = 2, \alpha = 0.25$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	1.0000	1.0000	1.0000	0.5315	0.5315	0.5315	1.25s
daCCIB	1.0000	1.0000	1.0000	0.5315	0.5315	0.5315	0.15s
CondEns[k-means]	0.7564	0.7780	0.7320	**0.7178**	0.7415	**0.6670**	0.02s
CondEns[EM]	0.7517	0.7790	0.7055	0.7141	**0.7540**	0.6130	0.02s
CondEns[daIB]	0.7230	0.9100	0.5670	0.6124	0.7075	0.5425	0.42s

Table 8.38: Results for binary synthetic sets: 50 randomly generated data sets with $K = 3, N = 200, M = 400, \alpha = 1.00$. Algorithms are run with 10 random initializations for each data set. Results are shown for the mean, max and min precision over initializations, averaged over all 10 sets. Highest scores are in bold.

$K = 3, \alpha = 1.00$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.5088	0.6091	0.3707	0.6484	0.8716	0.4962	7.43s
daCCIB	0.4700	0.4933	0.4476	**0.8487**	0.8624	**0.8120**	0.66s
CondEns[k-means]	0.5000	0.5307	0.4493	0.6544	0.7244	0.5940	0.07s
CondEns[EM]	0.4598	0.5144	0.4191	0.7826	**0.8784**	0.6560	0.08s
CondEns[daIB]	0.4600	0.5149	0.4222	0.7863	0.8660	0.6509	0.83s

Table 8.39: Results: 50 randomly generated data sets with orthogonality weight $\alpha = 0.75$.

$K = 3, \alpha = 0.75$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.5916	0.6636	0.5176	**0.6066**	**0.7713**	0.5162	9.24s
daCCIB	0.5838	0.5904	0.5771	0.5562	0.5573	**0.5551**	0.68s
CondEns[k-means]	0.5659	0.5876	0.5336	0.5586	0.5749	0.5302	0.07s
CondEns[EM]	0.5569	0.5716	0.5396	0.5672	0.5833	0.5433	0.07s
CondEns[daIB]	0.5447	0.6207	0.4656	0.5326	0.5869	0.4564	0.90s

Table 8.40: Results: 50 randomly generated data sets with orthogonality weight $\alpha = 0.50$.

$K = 3, \alpha = 0.50$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.7815	0.8000	0.7462	0.4812	0.5462	0.4500	10.75s
daCCIB	0.7721	0.7842	0.7631	0.4620	0.4704	0.4518	0.62s
CondEns[k-means]	0.6574	0.7842	0.4809	0.5172	0.5738	0.4482	0.07s
CondEns[EM]	0.6682	0.7718	0.5153	**0.5250**	**0.5860**	**0.4544**	0.07s
CondEns[daIB]	0.5041	0.6093	0.4096	0.4608	0.5102	0.4000	0.93s

Table 8.41: Results: 50 randomly generated data sets with orthogonality weight $\alpha = 0.25$.

$K = 3, \alpha = 0.25$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.9996	0.9996	0.9996	0.3640	0.3640	0.3640	5.48s
daCCIB	0.9993	0.9993	0.9993	0.3638	0.3638	0.3638	0.51s
CondEns[k-means]	0.6548	0.7936	0.4771	0.4685	0.5102	**0.4238**	0.08s
CondEns[EM]	0.6581	0.7864	0.5091	**0.4695**	**0.5142**	0.4193	0.07s
CondEns[daIB]	0.5286	0.6222	0.4216	0.4181	0.4560	0.3673	0.91s

Table 8.42: Results for binary synthetic sets: 50 randomly generated data sets with $K = 4, N = 200, M = 400, \alpha = 1.00$. Algorithms are run with 10 random initializations for each data set. Results are shown for the mean, max and min precision over initializations, averaged over all 10 sets. Highest scores are in bold.

$K = 4, \alpha = 1.00$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.4443	0.4959	0.3964	0.5883	0.7007	0.4664	28.81s
daCCIB	0.4414	0.4585	0.4322	0.6503	0.6806	**0.5867**	1.47s
CondEns[k-means]	0.3842	0.4157	0.3559	0.5611	0.5974	0.5130	0.40s
CondEns[EM]	0.3491	0.3827	0.3197	**0.6823**	**0.7688**	0.5733	0.28s
CondEns[daIB]	0.3796	0.4224	0.3355	0.5536	0.6596	0.4339	1.86s

Table 8.43: Results: 50 randomly generated data sets with orthogonality weight $\alpha = 0.75$.

$K = 4, \alpha = 0.75$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.4943	0.5167	0.4597	0.4996	**0.5661**	0.4049	29.37s
daCCIB	0.5041	0.5060	0.5036	**0.5017**	0.5019	**0.5012**	1.01s
CondEns[k-means]	0.4582	0.4853	0.4363	0.4345	0.4553	0.4076	0.41s
CondEns[EM]	0.4294	0.4456	0.3990	0.4600	0.4898	0.4236	0.31s
CondEns[daIB]	0.3832	0.4193	0.3456	0.4045	0.4615	0.3486	2.09s

Table 8.44: Results: 50 randomly generated data sets with orthogonality weight $\alpha = 0.50$.

$K = 4, \alpha = 0.50$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.6066	0.6286	0.5927	**0.4199**	**0.4524**	0.3737	43.14s
daCCIB	0.6119	0.6182	0.6032	0.4058	0.4219	**0.3919**	1.17s
CondEns[k-means]	0.5882	0.6733	0.4955	0.4028	0.4421	0.3611	0.37s
CondEns[EM]	0.5387	0.6145	0.4594	0.4051	0.4510	0.3712	0.35s
CondEns[daIB]	0.4076	0.4741	0.3569	0.3875	0.4295	0.3458	1.99s

Table 8.45: Results: 50 randomly generated data sets with orthogonality weight $\alpha = 0.25$.

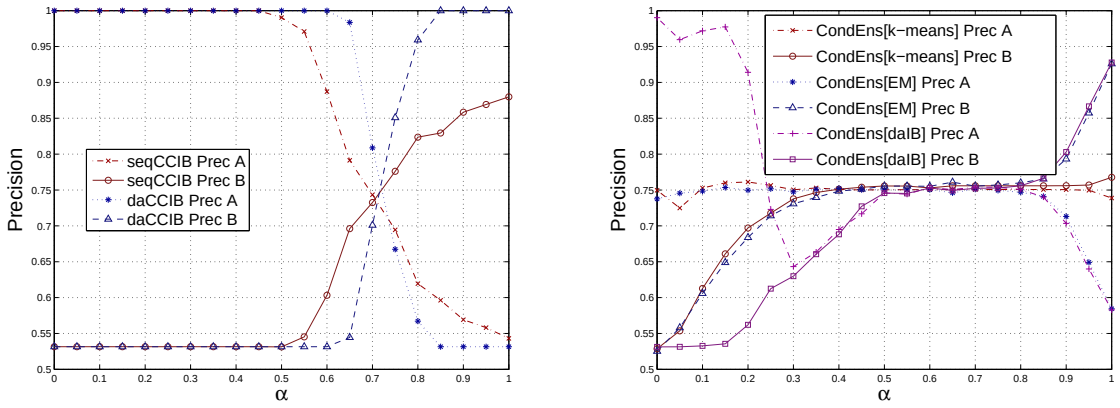
$K = 4, \alpha = 0.25$							
Algorithm	$Prec_A(C)$			$Prec_B(C)$			runtime
	mean	max	min	mean	max	min	mean
seqCCIB	0.9849	0.9860	0.9832	0.2759	0.2765	0.2754	22.97s
daCCIB	0.9772	0.9819	0.9752	0.2761	0.2782	0.2739	1.06s
CondEns[k-means]	0.6384	0.7800	0.5172	0.3463	**0.3852**	0.3183	0.28s
CondEns[EM]	0.6282	0.7694	0.5340	**0.3485**	0.3805	**0.3209**	0.29s
CondEns[daIB]	0.4031	0.4706	0.3488	0.3285	0.3503	0.2961	1.57s

however, obtains a substantially better maximum score of 1.000 versus daCCIB's max of 0.9080 and a substantially worse minimum score of 0.5230 versus daCCIB's minimum score of 0.8365. For $K = 3$ [Table 8.39], seqCCIB has a higher mean score with 0.6066 than daCCIB which obtains 0.5562. In fact, daCCIB performs poorly overall, scoring a max $Prec_B(C)$ of only 0.5573 versus daCCIB's max of 0.7713. Examining the plot in Figure 8.9(a), it becomes evident that there is a range of α (specifically, $\alpha = 0.6$ to $\alpha = 0.85$), for which seqCCIB outperforms daCCIB. The outperformance of the sequential method is not evident for $K = 4$ [Table 8.43] where at $\alpha = 0.75$, the mean performance of the algorithms is roughly equal (seqCCIB scores 0.4996 and daCCIB scores 0.5017.) Of course, the sequential approach does obtain a higher max score versus the deterministic annealing max (0.5661 vs. 0.5019.) However examining the plot in Figure 8.9(a), it is apparent that there is not a range of α with outperformance by seqCCIB as there was for $K = 3$. It is difficult to detect any general properties of the algorithms with respect to α across K other than the observation that daCCIB does consistently outperform seqCCIB for high α where its mean precision is higher than seqCCIB when $\alpha > 0.70$ for $K = 2$, $\alpha > 0.85$ for $K = 3$, and $\alpha > 0.8$ for $K = 4$. For lower α, the experiments do not consistently favor daCCIB or seqCCIB in all choices of K.

At lower α and for all K, the CCIB algorithms perform poorly, obtaining solutions more similar to the known partition A than target partition B. Again, this is an issue which arises due to the coordination term. Sets with lower α share the characteristic that the undesired clustering A is a much higher quality clustering than B when measured over the entire set. This affects CCIB directly as the coordination term favors clusterings which are high quality with respect to the entire set.

Analysis of *CondEns* algorithms as orthogonality varies.

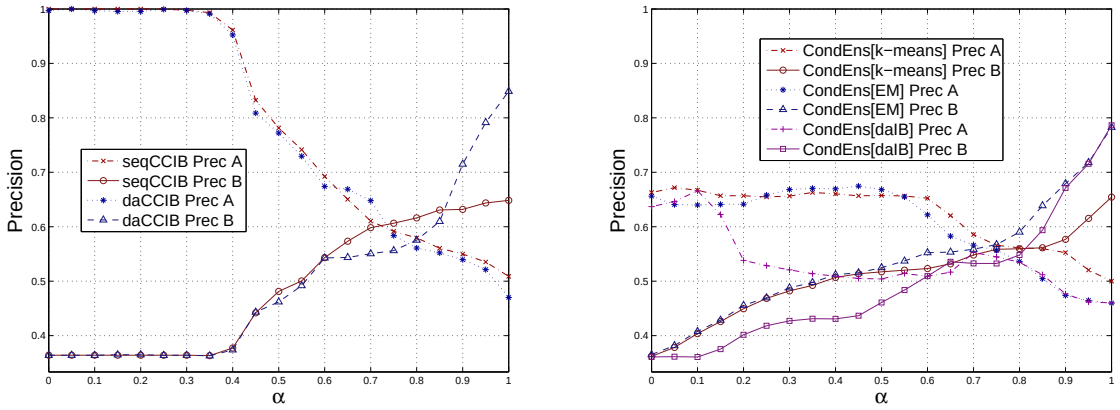
We now consider the performance of the *CondEns* algorithms on the data sets where $\alpha < 1.0$. In general, considering Tables 8.35 through 8.45, *CondEns[EM]* shows the best performance overall. For combinations with $\alpha = 0.75, 0.50, 0.25$ and $K = 2, 3, 4$, *CondEns[EM]* obtains the highest $Prec_B(C)$ in 7 out of the 9 combinations. Over all combinations, *CondEns[EM]* on average outscores



(a) CCIB algorithms

(b) *CondEns*

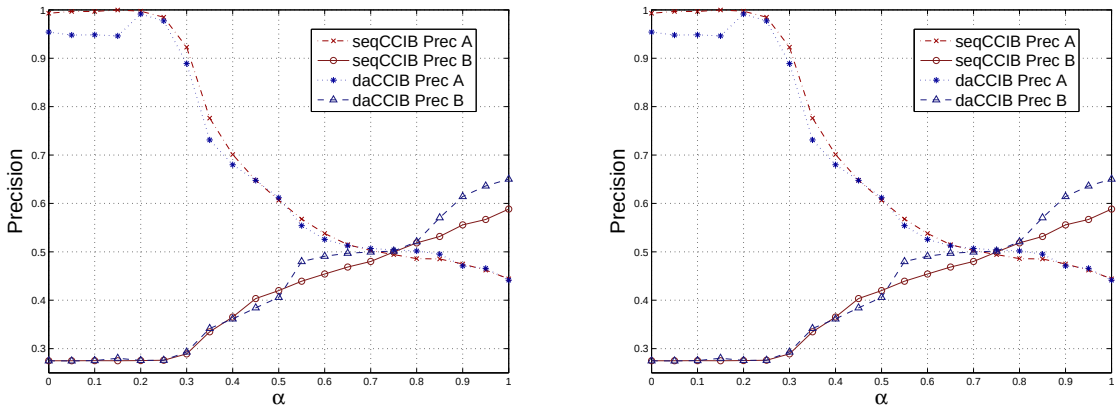
Figure 8.8: Mean precision scores for $K = 2$ where the orthogonality constant α varies.



(a) CCIB algorithms

(b) *CondEns*

Figure 8.9: Mean precision scores for $K = 3$ where the orthogonality constant α varies.



(a) CCIB algorithms

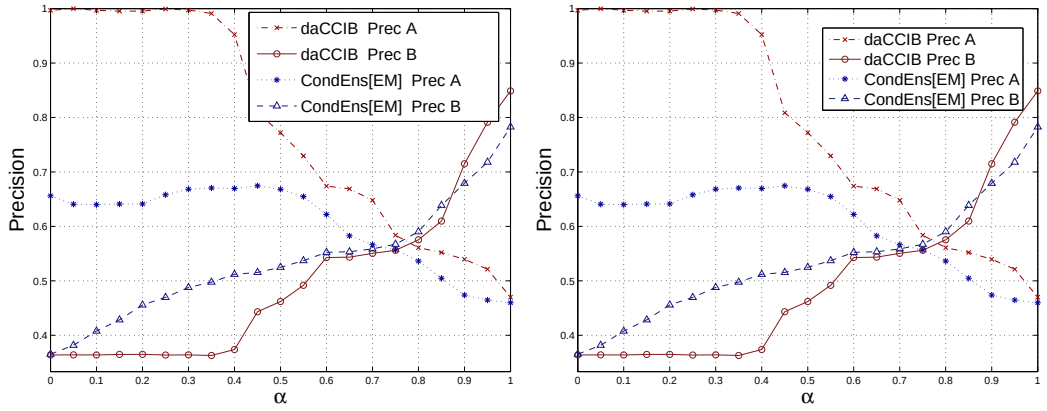
(b) *CondEns*

Figure 8.10: Mean precision scores for $K = 4$ where the orthogonality constant α varies.

CondEns[daIB] in mean precision by 0.0399 whereas it scores only 0.0050 higher than *CondEns[k-means]*. Examining Figures 8.8(a), 8.9(a), and 8.10(a) shows more precisely when *CondEns[EM]* has an advantage over the other two algorithms. Figure 8.8(a) shows for $K = 2$ the performance of *CondEns[EM]* and *CondEns[daIB]* is roughly equal for all α. For $K = 3$, however, Figure 8.9(a) shows that *CondEns[EM]* has a slight advantage over the next highest scoring, *CondEns[daIB]*, from $\alpha = 0.7$ to $\alpha = 0.9$. The advantage of *CondEns[EM]* is more clearly demonstrated in 8.10(a) at $K = 4$ where *CondEns[EM]* decidedly outscores the other two algorithms from $\alpha = 0.55$ to $\alpha = 1.0$. In conclusion, *CondEns[EM]* is the best overall performer in this setting.

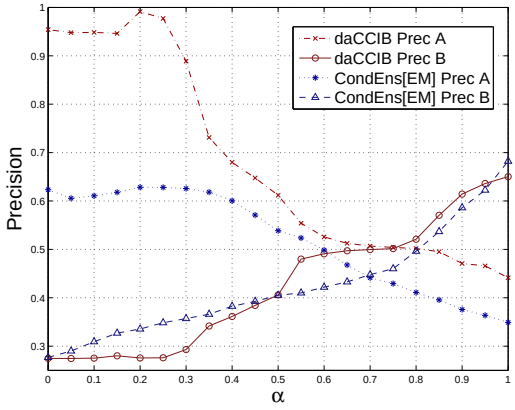
Comparison of CCIB and *CondEns* algorithms as orthogonality varies.

For easy comparison, the mean precision results for daCCIB and *CondEns[EM]* are plotted in Figure 8.11. The results show that *CondEns[EM]* on average underperforms CCIB for orthogonal sets (high α). For lower α, however, the results favor *CondEns* At $K = 2$, for instance, with $\alpha = 0.25$, the mean precision $Prec_B(C)$ of *CondEns[EM]* is 34.36% higher than that of daCCIB and the max precision is 41.86% higher than daCCIB. The advantage of *CondEns[EM]* does decrease as K increases however, with a mean increase of 29.05% at $\alpha = 0.25, K = 3$ and a mean increase of



(a) K=2

(b) K=3



(c) K=4

Figure 8.11: CCIB vs. *CondEns*: Mean precision scores for $K = 2$ where the orthogonality constant α varies.

26.22% at $\alpha = 0.25, K = 4$. Sets with lower α share the characteristic that the undesired clustering A is a much higher quality clustering than B when measured over the entire set. This affects CCIB directly as it has a coordination term that favors clusterings which are high quality with respect to the entire set. *CondEns*, on the other hand, finds high quality clusterings independently within the pre-image sets. In those cases, B is still a higher quality clustering.

8.6 Parameter sensitivity

8.6.1 Sensitivity of penalty-based methods

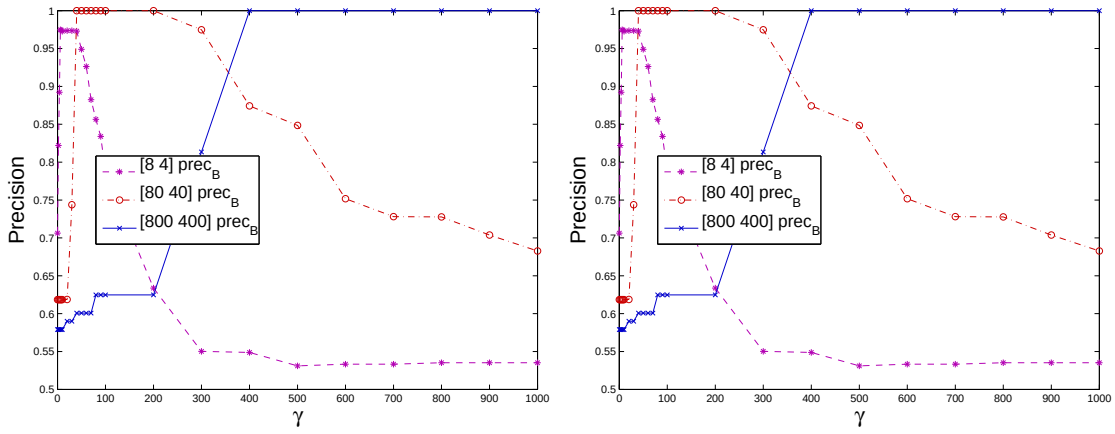
Both the Information Bottleneck with Side Information (IBwS) and Constrained Model-Based Clustering (CMBC) have objective functions which may be interpreted as balancing between a likelihood-like term and a redundancy penalty term. We review both objective functions in (8.6.1) and (8.6.2) below. In both cases there is a tuning parameter γ to trade off between the two terms.

$$IBwS: \quad \mathcal{F}_{IBwS} = \max\ I(C;Y) - \gamma I(C;Z) \tag{8.6.1}$$

$$CMBC: \quad \mathcal{L}_{CMBC} = \max\ l(Y) - \gamma H(Z|C) \tag{8.6.2}$$

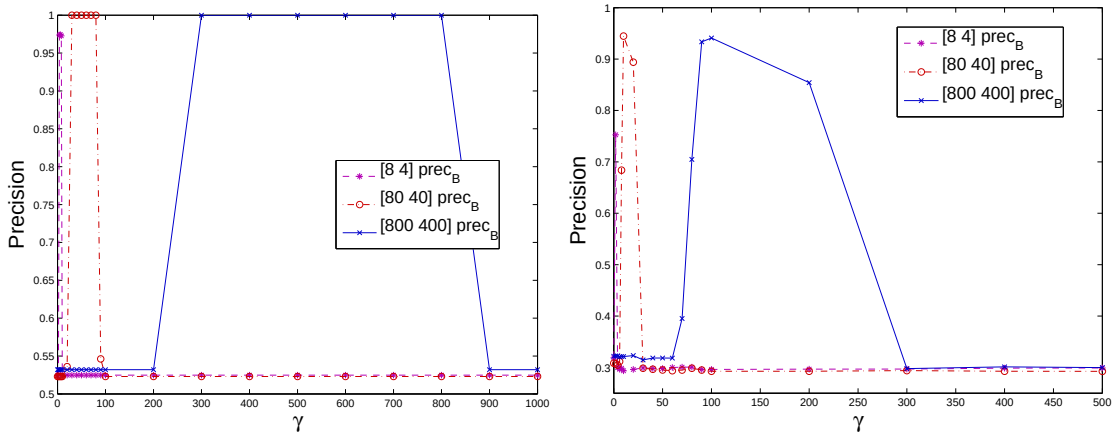
Synthetic sets

Experiments using the Information Bottleneck with Side Information suggested the solution found is dependent on the value for γ. The range of γ for which good solutions are found may in practice be somewhat narrow. We investigate the sensitivity of γ for synthetic test sets. Synthetic data sets are generated with $K = 2$ and $K = 4$. Experiments are performed for data sets where the dimensionality of Y is 12, 120, and 1200. Within the algorithm, we assume a Bernoulli distribution. As seen in Figure 8.12, the algorithm performs well for certain γ. For the higher-dimensional sets, the algorithm obtains the desired solution exactly, which is to be expected as the high dimensionality prevents any spurious clusterings obtained due to the random noise in the sets. For incorrect γ, however, the algorithm fails to find the desired solution. Further, the correct γ varies substantially



(a) seqIBwS, K=2

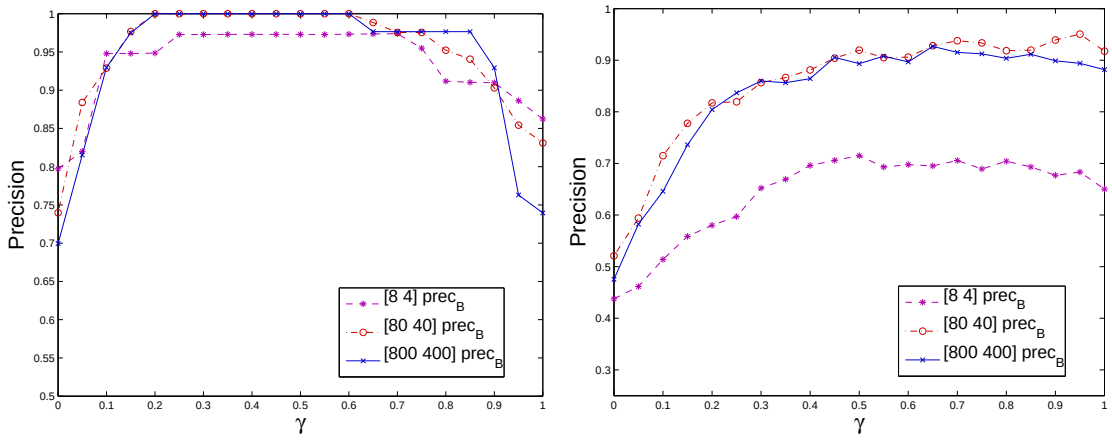
(b) seqIBwS, K=4



(c) daIBwS, K=2

(d) daIBwS, K=4

Figure 8.12: Varying γ in IBwS. Each line corresponds to data sets with dimensionality Y indicated in terms of $[m_A, m_B]$ in the legend. For different dimensionalities, the optimal γ do not overlap.



(a) daCCIB, K=2

(b) daIBwS, K=4

Figure 8.13: Varying γ in IBwS. Each line corresponds to data sets with dimensionality Y indicated in terms of $[m_A, m_B]$ in the legend. For different dimensionalities, the solutions are insensitive to γ and the same γ may be used for all dimensionalities.

over the data sets considered and there is no γ value which works for all sets. The performance of the sequential algorithm degrades more gracefully than that of the deterministic annealing approach. We conclude that the Information Bottleneck with Side Information requires particular settings of γ which vary according to the data set under consideration.

8.6.2 Sensitivity of CCIB coordination term

Synthetic sets

The CCIB formulation also has a free parameter λ. Unlike the Information Bottleneck with Side Information, the λ is used in the CCIB to control the weight accorded to the coordination term $I(C;Y)$. Typically, $I(C;Y|Z) \leq I(C;Y)$ and as we do not want the coordination term to outweigh the conditional term, we consider $0 \leq \lambda \leq 1$. We find in practice the CCIB is quite robust to a range of λ. Results on the synthetic sets are shown in Figure 8.13. For the different dimensionalities, the solutions obtained by the algorithm are insensitive with respect to γ. The same γ may be used over all cases and in practice we take $\gamma = 0.3$.

8.7 Discussion of results

Most importantly, throughout the experiments considered, the three approaches are able to extract the novel secondary structure. That is, they find solutions which are unlike the known clustering and typically similar to the unknown clustering. Though the performance depends on particulars of the data sets, this does serve as a general proof-of-concept that it is possible to perform non-redundant clustering.

Examining the relative performance of the algorithms on the various data sets, a number of trends become apparent. For those data sets which meet its requirements, CMBC often outperforms the other two approaches. The outperformance of CMBC, however, does come at a somewhat greater computational cost. A more significant drawback is the dependency of CMBC on a tuning parameter. While it is shown that this tuning parameter is less sensitive than the parameters of existing methods, it still requires some degree of tuning and this demand for tuning may be unreasonable in exploratory settings.

Over all the data sets considered, the *CondEns* approach frequently outperforms CCIB while at a fraction of the computational cost. It does not, however, strictly dominate CCIB in terms of average performance. Notably, on several text data sets, CCIB does outperform *CondEns*. A potential concern is that *CondEns* is motivated by rather strong demands on the data set, namely that the structures be information-orthogonal (the so-called "orthogonality assumption.") Notably, despite this strict theoretical requirement, *CondEns* performs robustly even as the orthogonality assumption is challenged, outperforming CCIB on data sets where the two clusterings are substantially correlated. A further advantage of *CondEns* is the lack of tuning parameters. This is not to ignore the fact that *CondEns* is parameterized by the base clustering method, which itself may require tuning parameters. As was shown in the experiments, typically performance was relatively consistent irregardless of which base clustering methods was employed, and no special tuning parameters were required in the base clustering methods. A potential disadvantage to *CondEns* is the

sensitivity to initialization. While its mean performance is better than CCIB, the range of solutions obtained typically contains solutions which are of lower quality than any of those obtained by CCIB.

In summary, all three techniques performed well throughout most of the experimental settings. Selecting the best-performing technique depends upon the data set and the particular setting. If the data is binary, clusters of similar size, and some parameter tuning is possible, CMBC outperforms the others. In other settings, *CondEns* often outperforms CCIB. CCIB, however, is less sensitive to initialization than *CondEns* and so the use of CCIB promises more consistent results.

Chapter 9

Extensions

In this chapter, two extensions to non-redundant clustering are discussed. First, in Section 9.2, the task of enumerating successive non-redundant clusterings is considered. One approach to performing this task relies on incorporating continuous known structure. This approach is compared against a categorical approach. Due to the difficulty of obtaining real-world data sets with many independent categorization schemes, focus is restricted to sample sets of synthetic data. As such, we consider this evaluation a proof-of-concept which may be further studied in the future as applications arise. Second, the case of semi-supervised non-redundant clustering is considered. In this work, we have until now assumed an exploratory, unsupervised setting, where there is no knowledge of desired solutions. In semi-supervised learning, it is assumed some supervised information exists (e.g. a portion of the instances are labeled according to a desired categorization scheme). In Section 9.3, we investigate whether semi-supervised learning can be enhanced by incorporating known undesired categorization schemes.

9.1 Continuous known structure

We have thusfar focused on categorical known structure. In Section 1.2 we stated that non-redundant clustering may be applied to non-categorical known structure. Both CMBC and CCIB may accept continuous known structure. CMBC accepts binary non-categorical known structure directly. CCIB accepts non-categorical known structure through the use of generalized linear models, as described in Algorithm 6.4. We will see the use of continuous known structure for the task of finding successive non-redundant clusterings.

9.2 Successive non-redundant clusterings

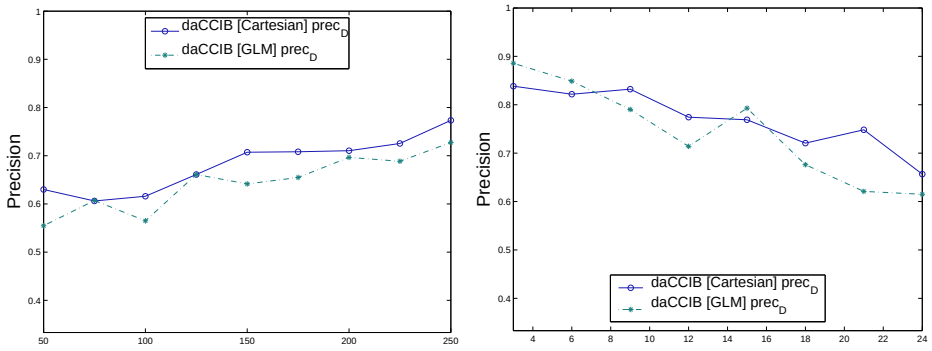
A natural problem which arises in interactive settings is, after finding a non-dominant clustering, to find the *next* non-dominant clustering. That is, how can this technique be generalized to enumerate arbitrarily many clusterings where each clustering is non-redundant with respect the clusterings that have come before? At issue is the modeling of the known structure which in this case consists of two or more clusterings. We consider two approaches to representing this information: the cartesian product of the known clusterings may be taken and then supplied as categorical Z to the algorithm or the concatenated membership vectors may be treated as a binary vector and then supplied as continuous Z to the algorithm.

One might expect the continuous approach to outperform the categorical. This is because by taking the cartesian product, one ignores the relation between different combinations within the cartesian product. For example, with known cluster memberships in A and B, no information is shared in estimating the Y distribution for an instance in combination (a_1, b_1) and an instance in combination (a_1, b_2). This is in contrast to the GLM approach which learns a function in Y over all (a, b) and so can share information between combinations.

We restrict our attention to CCIB and CMBC which both support the continuous approach. Either the continuous or categorical approach may be used with CCIB and we evaluate both in

Algorithm	$Prec_D$ mean	best	worst	time mean
CCIB, cartesian product	0.9300	0.9300	0.9300	0.36s
CCIB, GLM	0.7300	0.7300	0.7300	64.42s

Table 9.1: N = 200 instances, dimensions associated with clusterings A,B,D = [8 8 4]. We assume A and B are known. 5 random initializations



(a) (b)

Figure 9.1: Obtaining successive non-redundant clusterings. For data sets containing independent partitionings A, B, D, comparing categorical representation (using Cartesian product of A and B) versus continuous representation (concatenating A and B as binary vector). For (a.), dimensions $[m_A\ m_B\ m_D]$= [8 8 4], N varies, K = 3. For (b.), m_D=6, N=200, K=3.

the following section. Afterwards, we consider CMBC which assumes the known structure may be non-categorical and so we do not have a choice of approaches.

9.2.1 CCIB

We evaluate both approaches on binary synthetic data sets with three partitionings, A, B, and D. The sets are generated according to the procedure detailed in Appendix 8.3.2. Using the number of features associated with each partitioning, $m_A = 8, m_B = 8, m_D = 4$, we assume A and B are known and evaluate the ability of the approaches to discover partitioning D. Results are shown in Table 9.1. We further consider the approaches as the data sets vary according to size and dimension. Figures 9.1(a) and 9.1(b) show performance as the number of instances and relative strength of partitions vary. The results show for this variety of settings, the cartesian product approach is in fact

approximately as successful as the GLM approach while coming at a fraction of the computational cost. This rough equivalence holds up when the number of instances varies and when the relative strength of the desired clustering is varied.

9.2.2 CMBC

Known Clusterings

The first experiment evaluates the ability of the algorithms to find successive clusterings and is divided into three sessions. For Session 1, we assume that one clustering, A is known, for Session 2 we assume that A and B are known, and for Session 3, we assume that A, B and D are known. In each session we consider datasets with p_{noise} ranging from 0.1 to 0.3. The value of γ for each of the algorithms is optimized over 20 possible settings for baseline setting $p_{noise} = 0.1$ and this value is retained for all other p_{noise} settings. Results are shown in Table 9.2.

Uncovering Underlying Structure We first evaluate the algorithms over all three sessions for the baseline setting where $p_{noise} = 0.1$. Here we expect the best performance as this is the setting for which the algorithms have been tuned. Performance among the three algorithms is approximately the same. While the performance of CMBCbu at finding the next most prominent clustering lags somewhat in Session 1, where there is a single known clustering, it improves relative to the other two algorithms in Sessions 2 and 3, where there are multiple known clusterings. The lower score of CMBCbu at discovering B in Session 1 occurs because in several of the trials, it instead discovers D. This is not a failure of the algorithm; it is successfully accomplishing the task of discovering a novel clustering, however it is not discovering the next most prominent clustering. We will discuss this phenomenon further when looking at other noise settings.

Examining results for higher noise settings, we find the performance of dIBwS drops dramatically. For example, consider Session 2 with $p_{noise} = 0.20$. CMBCbu finds solutions that average $NMI(C, D) = 0.4782$, and CMBCpm that average $NMI(C, D) = 0.5007$ whereas for dIBwS,

Table 9.2: Mean NMI for 100 synthetic sets generated with 1000 instances according to the procedure in Appendix 8.3.2. Parameter settings are: $\alpha = 0.5$, $\gamma^{CMBCbu} = .97, \gamma^{CMBCpm} = .95, \gamma^{IBwS} = 2.5714$.

Session 1: A is known. Goal is discovery of B, D or E.

	Session 1: $\mathbf{z} = A$				
p_{noise}	Algorithm	$NMI(C, A)$	$NMI(C, B)$	$NMI(C, D)$	$NMI(C, E)$
0.10	**CMBCbu**	0.0007	0.8653	0.0591	0.0006
	CMBCpm	0.0008	0.9297	0.0008	0.0006
	dIBwS	0.0008	0.9072	0.0173	0.0007
0.20	**CMBCbu**	0.0007	0.6136	0.0531	0.0008
	CMBCpm	0.0007	0.6599	0.0163	0.0007
	dIBwS	0.0107	0.6595	0.0114	0.0006
0.30	**CMBCbu**	0.0006	0.2546	0.0553	0.0048
	CMBCpm	0.0006	0.2796	0.0279	0.0025
	dIBwS	1.0000	0.0008	0.0006	0.0008

Session 2: A and B are known. Goal is discovery of D or E.

	Session 2: $\mathbf{z} = A, B$				
p_{noise}	Algorithm	$NMI(C, A)$	$NMI(C, B)$	$NMI(C, D)$	$NMI(C, E)$
0.10	**CMBCbu**	0.0006	0.0008	0.8336	0.0009
	CMBCpm	0.0005	0.0006	0.8300	0.0008
	dIBwS	0.0006	0.0008	0.8235	0.0085
0.20	**CMBCbu**	0.0006	0.0008	0.4782	0.0498
	CMBCpm	0.0005	0.0007	0.5007	0.0317
	dIBwS	0.0108	0.8404	0.0833	0.0006
0.30	**CMBCbu**	0.0110	0.0009	0.1958	0.0205
	CMBCpm	0.0006	0.0006	0.1520	0.0283
	dIBwS	0.1407	0.8601	0.0008	0.0007

Session 3: A, B and D are known. Goal is discovery of E.

	Session 3: $\mathbf{z} = A, B, D$				
p_{noise}	Algorithm	$NMI(C, A)$	$NMI(C, B)$	$NMI(C, D)$	$NMI(C, E)$
0.10	**CMBCbu**	0.0009	0.0006	0.0006	0.8176
	CMBCpm	0.0008	0.0006	0.0005	0.7997
	dIBwS	0.0007	0.0006	0.0005	0.8068
0.20	**CMBCbu**	0.0009	0.0005	0.0006	0.5220
	CMBCpm	0.0009	0.0005	0.0005	0.4351
	dIBwS	0.0006	0.0208	0.9800	0.0006
0.30	**CMBCbu**	0.0086	0.0013	0.0068	0.2107
	CMBCpm	0.0047	0.0008	0.0101	0.1160
	dIBwS	0.0393	0.1985	0.7270	0.0006

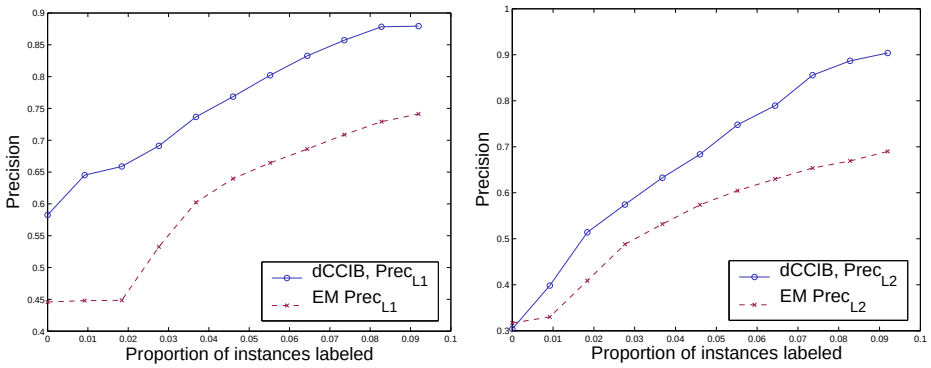
$NMI(C, D) = 0.0008$. The dIBwS algorithm is finding solutions similar to known clustering B. This behavior is consistent for the higher noise ($p_{noise} \geq 0.20$) settings throughout Sessions 2 and 3 where dIBwS fails to discover unknown clusterings. Interestingly, in these cases, dIBwS seems to consistently discover the weakest of the known clusterings, as can be seen by looking at Sessions 2 and 3. In Session 2, where B is the weakest known clustering, $NMI(C, B)$ is 0.8404 and 0.8601 for p_{noise} set to 0.20 and 0.30. In Session 3, where D is the weakest known clustering, $NMI(C, D)$ is 0.9800 and 0.7270 for p_{noise}set to 0.20 and 0.30. For all of these settings, the solutions obtained by CMBCbu and CMBCpm are most like the target clustering, whereas dIBwS largely fails to discover the target clustering.

In comparing CMBCbu and CMBCpm, there is not a clear winner. As we saw in the baseline setting of $p_{noise} = 0.10$, CMBCbu's performance relative to CMBCpm and dIBwS generally improves across sessions as there are more known clusterings. There does not, however, appear to be a clear trend within a given session as the noise is increased.

Finally, in Sessions 1 and 2, where there are multiple unknown clusterings, dIBwS almost always finds the next most prominent clustering whereas CMBCbu and CMBCpm occasionally discover less prominent clusterings (e.g. D and E in Sessions 1 and 2 in Table 9.2.) In general, the CMBC algorithms were more sensitive to initialization than dIBwS which often obtains the same solution regardless of initialization. This is despite the fact that all three algorithms are using a deterministic annealing framework.

9.3 Semi-supervised learning with background information

Up to this point, we have assumed that no supervised information, in the form of correctly-labeled instances, is available. We now address the question of whether semi-supervised learning may be enhanced by also avoiding redundancy with background knowledge. For example, one may wish to classify webpages according to a topic categorization. Typically in such settings, background



(a) L2 known

(b) L1 known

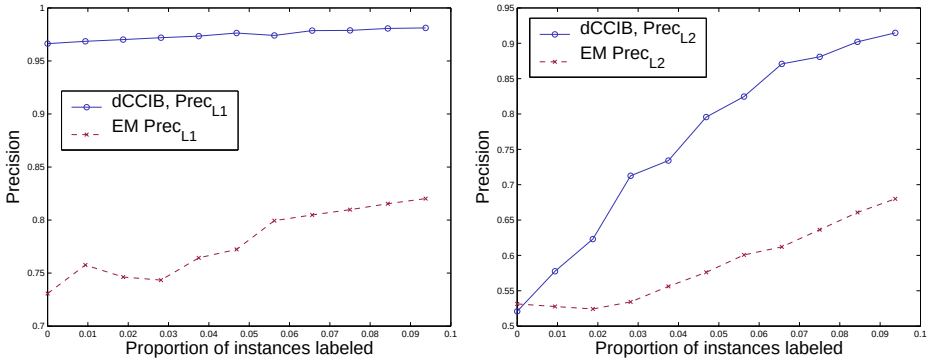
Figure 9.2: WebKB results: adding labeled data. Mean precision over 10 initializations. CCIB, which makes use of the known undesired categorization, outscores EM at finding the desired categorization.

Data set	session	Percentage improvement		
		$\alpha = 0$	$\alpha \approx .05$	$\alpha \approx .10$
WebKB	$Z = L1$	-4.07%	19.13%	31.02%
	$Z = L2$	30.69%	20.15%	18.62%
gmcat2x2	$Z = L1$	-1.95%	38.11%	34.51%
	$Z = L2$	32.25%	26.41%	19.63%
ec5x6	$Z = L1$	4.85%	19.35%	7.66%
	$Z = L2$	39.03%	25.19%	17.14%
top7x9	$Z = L1$	65.57%	42.27%	14.83%
	$Z = L2$	51.20%	42.54%	20.69%
average		16.80%	29.14%	20.51%

Table 9.3: Percent improvement in mean precision scores taken over range of labeled data proportions α

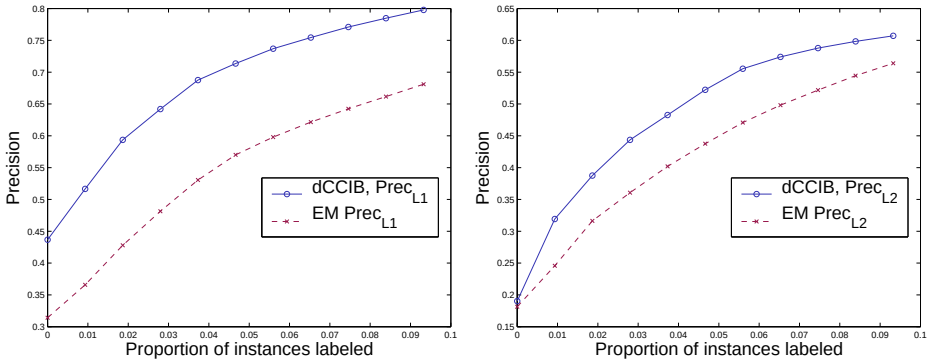
knowledge of other categorization schemes (e.g. URL) is discarded. Here, we ask whether background knowledge of undesired categorizations can in fact enhance classification along a desired categorization.

We assume a small set of examples is labeled according to a desired categorization. Performance is compared against EM augmented with a Naive Bayes model for labeled data, as described in [Nigam et al., 2000]. Labeled data is incorporated into the CCIB framework by fixing the corresponding $P(c|x_i)$ for each labeled instance x_i. Results for varying α, the proportion of labeled data, are shown in Figures 9.2, 9.3, 9.4, and 9.5. Equal numbers of instances from each target category are labeled. The mean precision over 10 random initializations is shown for each of the text data



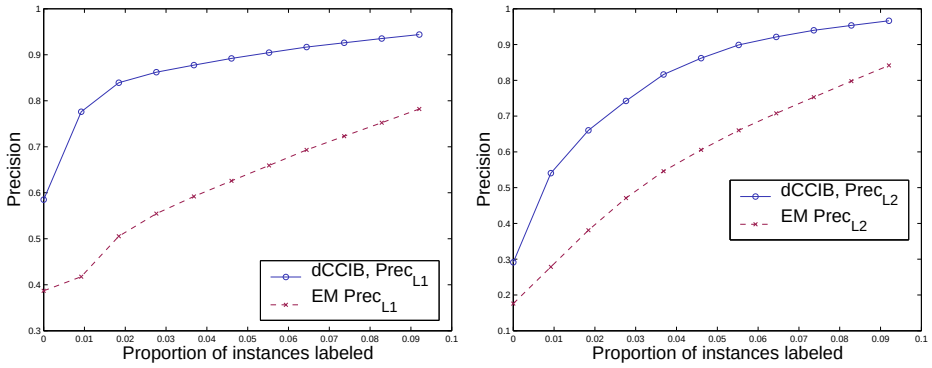
(a) L2 known (b) L1 known

Figure 9.3: RCV1-gmcat2x2 results: adding labeled data. Mean precision over 10 initializations. CCIB, which makes use of the known undesired categorization, outscores EM at finding the desired categorization.



(a) L2 known (b) L1 known

Figure 9.4: RCV1-ec5x6 results: adding labeled data. Mean precision over 10 initializations. CCIB, which makes use of the known undesired categorization, outscores EM at finding the desired categorization.



(a) L2 known

(b) L1 known

Figure 9.5: RCV1-top7x9 results: adding labeled data. Mean precision over 10 initializations. CCIB, which makes use of the known undesired categorization, outscores EM at finding the desired categorization.

sets. The results show that using background information about undesired clusterings can significantly improve the classification performance. The performance gain varies with the proportion of instances labeled. Table 9.3 lists the average, minimum and maximum performance gains obtained for each data set. For the regime considered, where the proportion of labeled data ranges from 0 to 10%, the CCIB method substantially outperforms EM. For proportion $\alpha = 0.05$ of the data labeled, there is an average performance improvement of 29.14%. For $\alpha = 0.10$, the average improvement is 20.51%.

9.4 Conclusions

In this chapter, non-redundant clustering was extended to enumerate successive non-redundant clusterings and to combine with supervised information to enhance semi-supervised clustering. In considering successive non-redundant clusterings, we established that enumeration of successive non-redundant clusterings is possible. With the coordinated conditional information bottleneck (CCIB), in simple experiments considering the two approaches, we found the cartesian product approach actually outperformed the continuous approach. With constrained model-based clustering (CMBC),

we showed that it is successful at enumerating multiple structures. We have focused on simple synthetic data sets in order to establish the feasibility of enumerating successive non-redundant clusterings. It is an open line of research to expand on these techniques in real-world settings.

In application of non-redundant clustering to semi-supervised learning, we found that known undesired categorization schemes can substantially improve the results of semi-supervised clustering. This is a powerful result as it offers a means by which to make use of information which would have otherwise been traditionally discarded. This information may be actually quite common in some settings such as webpage clustering, where URL information can be interpreted as a categorization scheme. The notable gains in performance on this task provide a convincing argument for further consideration of the combination of non-redundant clustering and supervised approaches.

Chapter 10

Conclusion

We have proposed the problem of non-redundant clustering, presented three distinct approaches to solving the problem, evaluated their performance on synthetic and real data sets, and discussed extensions to related problems. Intuitively, the goal of non-redundant clustering is to obtain a clustering which is novel, or non-redundant, with respect to some given known structure. Using relevant concepts from information theory, we introduced the notion of information-orthogonal clusterings which is then used as a key constraint in the formal problem definition. The definition given is sufficiently general to apply to categorical or continuous known structure.

In Chapter 3, an overview of several widely-used clustering algorithms as well as more recent algorithmic innovations were given. These algorithms provide the foundation and inspiration for the three approaches we develop in Chapters 5, 6, and 7. Table 10.1 compares properties of the algorithms.

Chapter 5 derived a model-based approach, CMBC, with a constrained likelihood objective function. The constraints are designed to handle approximately equal-sized clusters. In order to preserve tractability, the data is assumed to be binary, however this assumption allows us to derive efficient Expectation Maximization algorithms. Furthermore, the binary assumption is well-suited to cases with categorical known structure, which is the main focus of this work. One potential

drawback to CMBC is its reliance on a tuning parameter which balances between the likelihood and penalty (conditional entropy) terms. While we have found performance to be relatively robust with respect to this parameter, optimal values are dependent on the data set.

Chapter 6 derived an approach based on maximizing a conditional mutual information score. An Information Bottleneck approach, which we dub Conditional Information Bottleneck (CIB) is derived. As stated, the CIB does not enforce coordination across the known structure and so we propose an enhanced Coordinated Conditional Information Bottleneck (CCIB) and a deterministic annealing approach to obtaining solutions. The resulting algorithm is capable of handling categorical or continuous known structure and a wide variety of distribution-based assumptions for Y, is insensitive to initialization, and requires no tuning parameter to trade off between cluster quality and redundancy. The approach to coordination does introduce a tuning parameter, however we find performance to be quite insensitive to this parameter when the data set contains strong, highly orthogonal clusterings.

Chapter 7 presents *CondEns*, a framework which makes use of clustering ensembles. The framework makes use of base clustering techniques which are applied to a subset of the data selected according to the known structure. A wide variety of general or domain-specific base clustering techniques may be used, which gives this framework a broad applicability. Furthermore, the algorithm is efficient and typically requires less computation than if the base clustering technique were applied to the entire data set. Another benefit is that no tuning parameter is required. In practice, this approach achieves performance competitive with the other techniques, with the only drawback that results are more sensitive to initialization.

In Chapter 8, we experimentally evaluate the algorithms. The results show that the algorithms successfully discover novel, non-redundant structure over a variety of synthetic and real data sets and settings. In Chapter 9, we consider two main extensions: successive non-redundant clustering and semi-supervised classification. Successive non-redundant clustering attempts to enumerate a sequence of non-redundant clusterings while semi-supervised classification seeks to make use of

Table 10.1: Comparison of Non-Redundant Clustering Algorithms presented.

	CMBC	CCIB	*CondEns*
Algorithm	Expectation Maximization (EM)	Information Bottleneck (IB)	Clustering Ensembles
Categorical known structure	yes	yes	yes
Continuous known structure	yes (if binary)	yes (using GLMs)	no
Distribution-based clustering	yes (Bernoulli)	yes	yes
Distance-based clustering	no	no	yes
Tuning parameters	yes	yes (for co-ordination)	no
Successive non-redundant clustering	yes (cross-product or continuous)	yes (continuous)	yes (cross-product)

small amounts of user-labeled data. We show that two of the algorithms, CMBC and CCIB, are able to successfully enumerate successive non-redundant clusterings in synthetic sets. Evaluating success on real data sets is problematic, however, as it is difficult to obtain data sets with arbitrarily many labeled clusterings, and it is difficult to evaluate the structure obtained on arbitrary data sets. We then consider the application to semi-supervised classification with the presence of known structure. This is a setting which can occur often in practice, e.g. when clustering webpages, often the URLs will provide a somewhat independent background categorization. Typically these background categorizations are ignored, however here we ask if they can be used to improve results. Results show that in fact a significant performance gain may be achieved by incorporating this known structure, which extends the usefulness of our suite of techniques from unsupervised settings to semi-supervised settings.

In summary, clustering is used to discover a grouping of objects in data. While it has been a well-known complication that data often contains multiple quality clusterings. existing work has largely concerned itself with tailoring objective functions to, or requiring the user to specify properties of, a desired clustering. We have formally derived non-redundant clustering approaches which allow a user to specify what is not desired. This lends itself naturally to an interactive, exploratory clustering approach as well as a systematic method to enumerate multiple clusterings in a data set. The techniques we have presented for accomplishing this task handle a broad range of settings.

Furthermore, they are based on frequently used and well-studied clustering techniques which allow the leverage of a large body of research and practical experience.

Bibliography

[Bezdek and Pal, 1998] Bezdek, J. C. and Pal, N. R. (1998). Some new indexes of cluster validity. *IEEE transactions on systems, man and cybernetics*, 28(3):301–315.

[Bilenko et al., 2004] Bilenko, M., Basu, S., and Mooney, R. J. (2004). Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the Twenty-first International Machine Learning Conference*.

[Bottou and Bengio, 1995] Bottou, L. and Bengio, Y. (1995). Convergence properties of the K-means algorithms. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems*, volume 7, pages 585–592.

[Bradley and Fayyad, 1998] Bradley, P. and Fayyad, U. (1998). Refining initial points for k-means clustering. In *The Fifteenth International Conference on Machine Learning*, pages 91–99.

[Cardano, 1545] Cardano, G. (1545). *Ars Magna*. Nurnberg.

[Chechik, 2003] Chechik, G. (2003). *An Information Theoretic Approach to the Study of Auditory Coding*. The Hebrew University.

[Chechik and Tishby, 2002] Chechik, G. and Tishby, N. (2002). Extracting relevant structures with side information. In *Advances in Neural Information Processing Systems 15*. MIT Press.

[Cohen and Singer, 1996] Cohen, W. W. and Singer, Y. (1996). Context-sensitive learning methods for text categorization. In Frei, H.-P., Harman, D., Schäuble, P., and Wilkinson, R., editors,

Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval, pages 307–315, Zürich, CH. ACM Press, New York, US.

[Corless et al., 1993] Corless, R. M., Gonnet, G. H., Hare, D. E. G., and Jeffrey, D. J. (1993). On Lambert's W function. preprint.

[Cover and Thomas, 1991] Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. John Wiley & sons.

[Cox and Reid, 1987] Cox, D. R. and Reid, N. (1987). Parameter orthogonality and approximate conditional inference. In *Journal of the Royal Statistical Society. Series B*, volume 49 of *1*, pages 1–39.

[Craven et al., 1998] Craven, M., DiPasquo, D., Freitag, D., McCallum, A. K., Mitchell, T. M., Nigam, K., and Slattery, S. (1998). Learning to extract symbolic knowledge from the World Wide Web. In *Proc. of the 15th Conf. of the American Association for Artificial Intelligence*, pages 509–516.

[Davidson and Satyanarayana, 2003] Davidson, I. and Satyanarayana, A. (2003). Speeding up k-means clustering by bootstrap averaging. In *Third IEEE International Conference on Data Mining, Workshop on Clustering Large Data Sets*.

[Dempster et al., 1977] Dempster, A., Laird, N., and Rubin, D. B. (1977). Maximum likelihood from incomplete data. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

[Dom, 2002] Dom, B. (2002). An information-theoretic external cluster-validity measure. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence*, pages 137–145.

[Forgy, 1965] Forgy, E. W. (1965). Cluster analysis of multivariate data: efficiency versus interpretability of classifications. In *Biometrics*, volume 21, pages 768–769.

[Fowlkes and Mallows, 1983] Fowlkes, E. B. and Mallows, C. L. (1983). A method for comparing two hierarchical clusterings. In *Journal of the American Statistical Association*, number 78 in 383, pages 553–569.

[Friedman et al., 2001] Friedman, N., Mosenzon, O., Slonim, N., and Tishby, N. (2001). Multivariate information bottleneck. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI '01)*.

[Gluck and Corter, 1985] Gluck, M. and Corter, J. E. (1985). Information, uncertainty, and the utility of categories. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, pages 283–287.

[Gondek and Hofmann, 2003] Gondek, D. and Hofmann, T. (2003). Conditional information bottleneck clustering. In *3rd IEEE International Conference on Data Mining, Workshop on Clustering Large Data Sets*.

[Gondek and Hofmann, 2004] Gondek, D. and Hofmann, T. (2004). Non-redundant data clustering. In *Proceedings of the 4th IEEE International Conference on Data Mining*.

[Gondek and Hofmann, 2005] Gondek, D. and Hofmann, T. (2005). Non-redundant clustering with conditional ensembles. In *submission*.

[Gondek et al., 2005] Gondek, D., Vaithyanathan, S., and Garg, A. (2005). Clustering with model-level constraints. In *SIAM International Conference on Data Mining*.

[Gordon, 1996] Gordon, A. (1996). A survey of constrained classification. *Computational Statsistics and Data Analysis*, 21:17–29.

[Halkidi et al., 2001] Halkidi, M., Batistakis, Y., and Vazirgiannis, M. (2001). On clustering validation techniques. *Journal on Intellicesn Information Systems*, 17(2-3):107–145.

[Harremoës and Topsøe, 2001] Harremoës, P. and Topsøe, F. (2001). Details for inequalities between entropy and index of coincidence derived from information diagrams. *IEEE Transactions on Information Theory*, 47:2944–2960.

[Hastie et al., 2001] Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer-Verlag, New York.

[Havrda and Charvát, 1967] Havrda, J. and Charvát, F. (1967). Quantification method of classification processes. Concept of structural a-entropy. *Kybernetika*, 3:30–35. Review by I. Csiszár in MR, vol. 34, no.8875.

[Hertz et al., 2004] Hertz, T., Bar-Hillel, A., and Weinshall, D. (2004). Boosting margin based distance functions for clustering. In *Twenty-first International Conference on Machine learning*. ACM Press.

[Hubert and Schultz, 1976] Hubert, L. and Schultz, J. (1976). Quadratic assignment as a general data-analysis strategy. In *British Journal of Mathematical and Statistical Psychologie*, volume 29, pages 190–241.

[Jaccard, 1912] Jaccard, P. (1912). The distribution of flora in the alpine zone. In *New Phytologist*, volume 11, pages 37–50.

[Jain and Dubes, 1988] Jain, A. K. and Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall.

[Jancey, 1966] Jancey, R. C. (1966). Multidimensional group analysis. In *Australian Journal of Botany*, number 1 in 14, pages 127–130.

[Jordan and Jacobs, 1993] Jordan, M. I. and Jacobs, R. A. (1993). Hierarchical mixtures of experts and the EM algorithm. Technical Report AIM-1440.

[Klein et al., 2002] Klein, D., Kamvar, S., and Manning, C. (2002). From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proceedings of the 19th International Conference on Machine Learning*.

[Kuhn, 1955] Kuhn, H. W. (1955). The hungarian method for the assignment problem. In *Naval Research Logistic Quarterly*, volume 2, pages 83–97.

[Law et al., 2002] Law, M. H., Jain, A. K., and Figueiredo, M. A. T. (2002). Feature selection in mixture-based clustering. In *Advances in Neural Information Processing Systems 15 (NIPS-2002)*. MIT Press.

[Li and Yamanishi, 1997] Li, H. and Yamanishi, K. (1997). Document classification using a finite mixture model. In Cohen, P. R. and Wahlster, W., editors, *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 39–47, Somerset, New Jersey. Association for Computational Linguistics.

[Li et al., 2004] Li, T., Ma, S., and Ogihara, M. (2004). Entropy-based criterion in categorical clustering. In *Twenty-first International Conference on Machine Learning*. ACM Press.

[MacQueen, 1967] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley. University of California Press.

[McCallum and Nigam, 1998] McCallum, A. and Nigam, K. (1998). A comparison of event models for naive bayes text classification.

[Meilă, 2003] Meilă, M. (2003). Comparing clusterings. In *Conference on Computational Learning Theory*.

[Minaei-Bidgoli et al., 2004] Minaei-Bidgoli, B., Topchy, A., and Punch, W. (2004). Ensembles of partitions via data resampling. In *Proceedings of the International Conference on Information Technology: Coding and Computing*.

[Mirkin, 2001] Mirkin, B. (2001). Reinterpreting the category utility function. *Machine Learning*, 45(2):219–218.

[Neal and Hinton, 1998] Neal, R. and Hinton, G. (1998). A view of the em algorithm that justifies incremental, sparse, and other variants. In Jordan, M. I., editor, *Learning in Graphical Models*. Kluwer.

[Ng et al., 1998] Ng, R. T., Lakshmanan, L. V., Han, J., and Pang, A. (1998). Exploratory mining and pruning optimizations of constrained association rule. In *Proceedings of ACM SIGMOD*, pages 13–24.

[Nigam et al., 1998] Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. M. (1998). Learning to classify text from labeled and unlabeled documents. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, pages 792–799, Madison, US. AAAI Press, Menlo Park, US.

[Nigam et al., 2000] Nigam, K., McCallum, A. K., Thrun, S., and Mitchell, T. M. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134.

[Pereira et al., 1993] Pereira, F. C. N., Tishby, N., and Lee, L. (1993). Distributional clustering of english words. In *Meeting of the Association for Computational Linguistics*, pages 183–190.

[Perez, 1977] Perez, A. (1977). ϵ-admissible simplifications of the dependence structure of a set of random variables. *Kybernetika*, 13:439–449.

[Rand, 1971] Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. In *Journal of the American Statistical Association*, pages 846–850.

[Rose, 1998] Rose, K. (1998). Deterministic annealing for clustering, compression, classification, regression, and related optimization problems.

[Schultz and Joachims, 2003] Schultz, M. and Joachims, T. (2003). Learning a distance metric from relative comparisons. In *Advances in Neural Information Processing Systems*.

[Slonim, 2002] Slonim, N. (2002). *The Information Bottleneck: Theory and Applications*. The Hebrew University.

[Slonim et al., 2002] Slonim, N., Friedman, N., and Tishby, N. (2002). Unsupervised document classification using sequential information maximization.

[Slonim and Tishby, 2000] Slonim, N. and Tishby, N. (2000). Agglomerative information bottleneck. In *Proc. of NIPS-12, 1999*, pages 617–623. MIT Press.

[Slonim and Weiss, 2002] Slonim, N. and Weiss, Y. (2002). Maximum likelihood and the information bottleneck. In *Advances in Neural Information Processing Systems*.

[Strehl and Ghosh, 2002] Strehl, A. and Ghosh, J. (2002). Cluster ensembles: A knowledge reuse framework for combining partitionings. *Journal of Machine Learning Research*, 3:583–617.

[Studený, 1989] Studený, M. (1989). *The concept of multiinformation in probabilistic decision making*. Czechoslovak Academy of Sciences, Prague.

[Tishby et al., 1999] Tishby, N., Pereira, F. C., and Bialek, W. (1999). The information bottleneck method. In *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377.

[Topchy et al., 2003] Topchy, A., Jain, A. K., and Punch, W. (2003). Combining multiple weak clusterings. In *IEEE International Conference on Data Mining*.

[Topsøe, 2003] Topsøe, F. (2003). Entropy and index of coincidence, lower bounds. preprint.

[Vaithyanathan and Dom, 1999a] Vaithyanathan, S. and Dom, B. (1999a). Generalized model selection for unsupervised learning in high dimensions.

[Vaithyanathan and Dom, 1999b] Vaithyanathan, S. and Dom, B. (1999b). Model selection and document clustering. In *The Sixteenth International Conference on Machine Learning (ICML-99)*.

[Vaithyanathan and Gondek, 2002a] Vaithyanathan, S. and Gondek, D. (2002a). Clustering with informative priors. Technical report, IBM Almaden Research Center.

[Vaithyanathan and Gondek, 2002b] Vaithyanathan, S. and Gondek, D. (2002b). Partially supervised taxonomy generation – case in point for integration of structured and unstructured data. In *INFORMS Conference on Operations Research/ OR and the Internet*, San Jose, CA.

[Wagstaff and Cardie, 2000] Wagstaff, K. and Cardie, C. (2000). Clustering with instance-level constraints. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1103–1110.

[Wagstaff et al., 2001] Wagstaff, K., Cardie, C., Rogers, S., and Schroedl, S. (2001). Constrained k-means clustering with background knowledge. In *Proceedings of the 18th International Conference on Machine Learning*.

[Watanabi, 1960] Watanabi, S. (1960). Information theoretical analysis of multivariate correlation. *IBM Journal of research and development*, 4:66–81.

[Xing et al., 2002] Xing, E. P., Ng, A. Y., Jordan, M. I., and Russell, S. (2002). Distance metric learning, with application to clustering with side information. In *Advances in Neural Information Processing Systems*.

[Zhang, 2003] Zhang, B. (2003). Regression clustering. In *Third IEEE International Conference on Data Mining*.

Appendix A

Proofs for Non-Redundant Clustering Definition

A.1 Proof of Lemma 1

Lemma 1. *If C and Z are information-orthogonal w.r.t. X, then*

$$H(C, Z) = H(C) + H(Z). \tag{A.1.1}$$

Proof. We assume the feature representation of X contains all the information used for clustering and so restrict clusterings to be functions of the feature representation $C : \mathcal{D} \to \{1, \ldots, K\}$. As a result, for two clusterings C and Z, C will be conditionally independent of Z. That is, $C \perp\!\!\!\perp Z|X$ or equivalently,

$$I(C, Z|X) = 0. \tag{A.1.2}$$

We now prove the lemma. First, $I(C, Z; X)$ may be expanded as:

$$I(C, Z; X) = H(C, Z) - H(C, Z|X). \tag{A.1.3}$$

Using the definition of information-orthogonal in (4.1.1):

$$I(C, Z; X) = I(C; X) + I(Z; X) = H(C) - H(C|X) + H(Z) - H(Z|X). \tag{A.1.4}$$

Substituting (A.1.3) into (A.1.4) and rearranging terms:

$$H(C, Z) = H(C) + H(Z) - H(C|X) - H(Z|X) + H(C, Z|X) \tag{A.1.5}$$

$$= H(C) + H(Z) - I(C, Z|X) \tag{A.1.6}$$

$$= H(C) + H(Z), \tag{A.1.7}$$

using (A.1.2) in the final step.

□

A.2 Proof of Lemma 3

As in Appendix A.1, we use the assumption that the feature representation of X contains all the information used for clustering and so restrict clusterings to be functions of the feature representation. As a result,

$$P(Z|C, X) = P(Z|X). \tag{A.2.1}$$

Lemma 3. *If C and Z are information-orthogonal w.r.t. X, then $VI(C, Z)$ is maximal:*

$$VI(C, Z) = H(C) + H(Z). \tag{A.2.2}$$

Proof. The maximal value of $VI(C, Z)$ is $H(C) + H(Z)$ as given in (3.4.4), and follows from the definition of $VI(C, Z)$:

$$VI(C, Z) = H(C) + H(Z) - 2I(C, Z). \tag{A.2.3}$$

The maximal value is achieved for $I(C, Z) = 0$. We note a consequence:

$$0 = I(C, Z) \tag{A.2.4}$$

$$0 = H(C) + H(Z) - H(C, Z) \tag{A.2.5}$$

$$H(C, Z) = H(C) + H(Z). \tag{A.2.6}$$

which is a necessary and sufficient condition for information-orthogonality as given by Lemma 1. □

Appendix B

Proofs for Constrained Model-Based Clustering (CMBC)

B.1 Proof of Lemma 4: Computation of $p(\mathbf{z}_i|c_k)$

Lemma 4. *The $p(\mathbf{z}_i|c_k)$ may be expanded as:*

$$p(\mathbf{z}_i|c_k) = \frac{1}{M}\sum_{j}^{M}\sum_{y_{\cdot j}\in\{0,1\}} p(\mathbf{z}_i|y_{\cdot j})p(y_{\cdot j}|c_k). \tag{B.1.1}$$

Proof. We show that it is equivalent to marginalize over the individual *features* y_j using $p(y_j|c_k)$ rather than marginalizing over all possible feature *vectors* $\mathbf{y}_i$ and using $p(\mathbf{y}_i|c_k)$.

Let $Y_{\setminus j} = \{Y_1, \ldots Y_{j-1}, Y_{j+1}, \ldots, Y_M\}$. Using (5.1.22) and (5.1.26) with (5.1.27):

$$p(\mathbf{z}_h|c_k) = \sum_{\mathbf{y}_i \in Y} p(\mathbf{z}_h|\mathbf{y}_i)p(\mathbf{y}_i|c_k) \tag{B.1.2}$$

$$= \sum_{\mathbf{y}_i \in Y} \frac{1}{M} \sum_{j}^{M} p(\mathbf{z}_h|\mathbf{y}_{ij}) \prod_{l=1}^{M} p(\mathbf{y}_{il}|c_k) \tag{B.1.3}$$

$$= \frac{1}{M} \sum_{j}^{M} \sum_{\mathbf{y}_i \in Y_{\setminus j}} \sum_{\mathbf{y}_{ij} \in \{0,1\}} p(\mathbf{z}_h|\mathbf{y}_{ij})p(\mathbf{y}_{ij}|c_k) \prod_{l \neq j}^{M} p(\mathbf{y}_{il}|c_k) \tag{B.1.4}$$

$$= \frac{1}{M} \sum_{j}^{M} \left[\sum_{\mathbf{y}_{ij} \in \{0,1\}} p(\mathbf{z}_h|\mathbf{y}_{ij})p(\mathbf{y}_{ij}|c_k) \right] \cdot \left[\sum_{\mathbf{y}_i \in Y_{\setminus j}} \prod_{l \neq j}^{M} p(\mathbf{y}_{il}|c_k) \right] \tag{B.1.5}$$

$$= \frac{1}{M} \sum_{j}^{M} \sum_{\mathbf{y}_{ij} \in \{0,1\}} p(\mathbf{z}_h|\mathbf{y}_{ij})p(\mathbf{y}_{ij}|c_k), \tag{B.1.6}$$

as $\sum_{\mathbf{y}_i \in Y_{\setminus j}} \prod_{l \neq j}^{M} p(\mathbf{y}_{il}|c_k)$ must sum to 1. □

B.2 Derivation of Objective Function

As a consequence of the $P(Z|Y_j)$ being fixed, the noise features need not be considered when optimizing the log-likelihood:

Lemma 11. *Maximizing $l(\mathcal{Y})$ is equivalent to maximizing $l(\mathcal{D})$.*

Proof.

$$l(\mathcal{D}) = \sum_{k=1}^{K} \sum_{i:z(i)=k} \log p(\mathbf{z}_i|\mathbf{y}_i) + \log p(\mathbf{y}_i|c_k)p(c_k) \tag{B.2.1}$$

$$= \sum_{k=1}^{K} \sum_{i:z(i)=k} A_i + \log p(\mathbf{y}_i|c_k)p(c_k) \tag{B.2.2}$$

$$= A + l(\mathcal{Y}). \tag{B.2.3}$$

where the constant A may be ignored when optimizing the log-likelihood. □

B.3 Calculation of $\tilde{\mathrm{H}}(C|Z)$

Expanding the $p(\mathbf{z}_i|c_k)$ terms in the $\tilde{\mathrm{H}}^{\mathrm{l}}(C,Z)$ term from (5.1.39)

Recall that $\tilde{\mathrm{H}}^{\mathrm{l}}(C,Z)$ is defined in (5.1.19) in terms of $p(\mathbf{z}_i|c_k)$. The parameters for the model, however, are $p(\mathbf{y}_i|c_k)$. We now expand the $p(\mathbf{z}_i|c_k)$ according to (5.1.27) in order to obtain expressions explicitly in terms of $p(\mathbf{z}_i|c_k)$. We will need the following quantity which appears in $\tilde{\mathrm{H}}^{\mathrm{l}}(C,Z)$:

$$IC(Z,C) = \sum_{\mathbf{z}_i \in \mathcal{Z}} \sum_{k=1}^{K} p(\mathbf{z}_i, c_k) \cdot p(\mathbf{z}_i, c_k) \tag{B.3.1}$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} [p(c_k)p(\mathbf{z}_i|c_k)] \cdot [p(c_k)p(\mathbf{z}_i|c_k)]$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} \left[p(c_k) \frac{1}{M} \sum_{j=1}^{M} \sum_{y_{\cdot j} \in \{0,1\}} p(\mathbf{z}_i|y_{\cdot j}) p(y_{\cdot j}|c_k) \right] \cdot \left[p(c_k) \frac{1}{M} \sum_{l=1}^{M} \sum_{y_{\cdot l} \in \{0,1\}} p(\mathbf{z}_i|y_{\cdot l}) p(y_{\cdot l}|c_k) \right] \tag{B.3.2}$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} p(c_k)^2 \frac{1}{M^2} \left[\sum_{j=1}^{M} p_{ij}^1 \theta_{jk} + p_{ij}^0 (1 - \theta_{jk}) \right] \cdot \left[\sum_{l=1}^{M} p_{il}^1 \theta_{lk} + p_{il}^0 (1 - \theta_{lk}) \right] \tag{B.3.3}$$

$$= \sum_{\mathbf{z}_i \in \mathcal{Z}} \sum_{k=1}^{K} p(c_k)^2 \frac{1}{M^2} \sum_{j=1}^{M} \sum_{l=1}^{M} \left[\left(p_{ij}^1 - p_{ij}^0 \right) \left(p_{il}^1 - p_{il}^0 \right) \right. \tag{B.3.4}$$

$$\left. \cdot \, \theta_{jk}\theta_{lk} + \left(p_{il}^0 (p_{ij}^1 - p_{ij}^0) \right) \theta_{jk} + \left(p_{ij}^0 (p_{il}^1 - p_{il}^0) \right) \theta_{lk} + \left(p_{ij}^0 p_{il}^0 \right) \right]. \tag{B.3.5}$$

where we have substituted $p_{ij}^0 = p(\mathbf{z}_i|y_{\cdot j} = 0), p_{ij}^1 = p(\mathbf{z}_i|y_{\cdot j} = 1)$ and $\theta_{jk} = p(y_{\cdot j}|c_k)$ for simplification. To further simplify the expression, we introduce constants A_0, A_1, A_2 according to the definitions as first mentioned in (5.1.38).

$$A_2(j,l) = \sum_{i=1}^{N} \left(p(\mathbf{z}_i|y_{\cdot j} = 1) - p(\mathbf{z}_i|y_{\cdot j} = 0) \right) \left(p(\mathbf{z}_i|y_{\cdot l} = 1) - p(\mathbf{z}_i|y_{\cdot l} = 0) \right), \tag{B.3.6}$$

$$A_1(j) = \sum_{i=1}^{N} \sum_{l=1}^{M} p(\mathbf{z}_i|y_{\cdot l} = 0)(p(\mathbf{z}_i|y_{\cdot j} = 1) - p(\mathbf{z}_i|y_{\cdot j} = 0)), \tag{B.3.7}$$

$$A_0 = \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{l=1}^{M} p(\mathbf{z}_i|y_{\cdot j} = 0) p(\mathbf{z}_i|y_{\cdot l} = 0). \tag{B.3.8}$$

These definitions make use of the fact that the summation over $\mathbf{z}_i$ may be brought inside for the definitions of A_0, A_1, A_2. Substituting these constants produces:

$$IC(Z,C) = \sum_{k=1}^{K} p(c_k)^2 \frac{1}{M^2} \cdot \left\{ \sum_{j,l} A_2(j,l)\theta_{jk}\theta_{lk} + 2\sum_{j} A_1(j)\theta_{jk} + A_0 \right\}. \tag{B.3.9}$$

Then the expanded expression for $\tilde{\mathrm{H}}^{\mathrm{l}}(C,Z)$ is:

$$\tilde{\mathrm{H}}^{\mathrm{l}}(C,Z) = \left(\delta_{|\mathcal{Z}|} - \beta_{|\mathcal{Z}|} IC(C,Z)\right) \tag{B.3.10}$$

$$= \alpha_{|\mathcal{Z}|} - \beta_{|\mathcal{Z}|} \sum_{k=1}^{K} p(c_k)^2 \frac{1}{M^2} \cdot \left\{ \sum_{j,l} A_2(j,l)\theta_{jk}\theta_{lk} + 2\sum_{j} A_1(j)\theta_{jk} + A_0 \right\}. \tag{B.3.11}$$

B.3.1 Expanding the $p(\mathbf{z}_i|c_k)$ terms in the $H(Z)$ term of (5.1.39)

In a similar fashion, one may the expanded version of $\tilde{\mathrm{H}}^{\mathrm{u}}(Z)$ from (5.1.19). First, expanding $IC(Z)$:

$$IC(Z) = \sum_{i=1}^{N} p(\mathbf{z}_i) \cdot p(\mathbf{z}_i) \tag{B.3.12}$$

$$= \sum_{i=1}^{N} \left[\sum_{k=1}^{K} p(c_k) p(\mathbf{z}_i|c_k) \right] \cdot \left[\sum_{r=1}^{K} p(c_r) p(\mathbf{z}_i|c_r) \right]$$

$$= \sum_{i=1}^{N} \left[\sum_{k=1}^{K} p(c_k) \frac{1}{M} \sum_{j=1}^{M} \sum_{y_{\cdot j} \in \{0,1\}} p(\mathbf{z}_i|y_{\cdot j}) p(y_{\cdot j}|c_k) \right] \cdot \left[\sum_{r=1}^{K} p(c_r) \frac{1}{M} \sum_{l=1}^{M} \sum_{y_{\cdot l} \in \{0,1\}} p(\mathbf{z}_i|y_{\cdot l}) p(y_{\cdot l}|c_r) \right]$$

$$= \sum_{i=1}^{N} \frac{1}{M^2} \left[\sum_{k=1}^{K} p(c_k) \sum_{j=1}^{M} p_{ij}^1 \theta_{jk} + p_{ij}^0 (1-\theta_{jk}) \right] \cdot \left[\sum_{r=1}^{K} p(c_r) \sum_{l=1}^{M} p_{il}^1 \theta_{lr} + p_{il}^0 (1-\theta_{lr}) \right]$$

$$= \sum_{i=1}^{N} \frac{1}{M^2} \sum_{k=1}^{K} \sum_{r=1}^{K} p(c_k) p(c_r)$$

$$\cdot \left(\sum_{j=1}^{M} \sum_{l=1}^{M} \left[\left(p_{ij}^1 - p_{ij}^0\right) \left(p_{il}^1 - p_{il}^0\right) \theta_{jk}\theta_{lk} + \left(p_{il}^0 (p_{ij}^1 - p_{ij}^0)\right) \theta_{jk} + \left(p_{ij}^0 (p_{il}^1 - p_{il}^0)\right) \theta_{lk} + \left(p_{ij}^0 p_{il}^0\right) \right] \right),$$

which yields:

$$\tilde{\mathrm{H}}^{\mathrm{u}}(Z) = (\ln|\mathcal{Z}|) \cdot \left(1 - Q \sum_{k=1}^{K} \sum_{r=1}^{K} p(c_k) p(c_r) \frac{1}{M^2} \right.$$

$$\left. \cdot \left\{ \sum_{j,l} A_2(j,l)\theta_{jk}\theta_{lr} + 2\sum_{j} A_1(j)\theta_{jk} + A_0 \right\} - \frac{Q}{|\mathcal{Z}|} \right). \tag{B.3.13}$$

B.4 Derivation of Class-Conditional M-Step Equations

The critical points of (5.1.39) may by obtained as follows: First, the expanded versions given in (B.3.11) and (B.3.13) are substituted into (5.1.39). Next, the standard variational approximation for EM is applied [for details, see [Neal and Hinton, 1998]]. The resulting objective function is:

$$\begin{aligned}\tilde{\mathcal{L}} =&(1-\gamma)\sum_{i=1}^{N}\log\sum_{k=1}^{K}p(\mathbf{y}_i|c_k)p(c_k) \\ &+\gamma\left(\alpha_{|\mathcal{Z}|}-\beta_{|\mathcal{Z}|}\sum_{k=1}^{K}p(c_k)^2\frac{1}{M^2}\cdot\left\{\sum_{j,l}A_2(j,l)\theta_{jk}\theta_{lk}+2\sum_{j}A_1(j)\theta_{jk}+A_0\right\}\right) \\ &-\gamma\Bigg(\left(\ln|\mathcal{Z}|\right)\cdot\Bigg(1-Q\sum_{k=1}^{K}\sum_{r=1}^{K}p(c_k)p(c_r)\frac{1}{M^2}\cdot\left\{\sum_{j,l}A_2(j,l)\theta_{jk}\theta_{lr}+2\sum_{j}A_1(j)\theta_{jk}+A_0\right\} \\ &-\frac{Q}{|\mathcal{Z}|}\Bigg)\Bigg).\end{aligned} \tag{B.4.1}$$

The objective function in B.4.2 is now maximized for θ_{hs}. Taking the derivative:

$$\frac{\partial\mathcal{L}}{\partial\theta_{hs}} =(1-\gamma)\sum_{i=1}^{N}q(c_s|\mathbf{d}_i)\left(\frac{\mathbf{y}_{ij}}{\theta_{hs}}-\frac{1-\mathbf{y}_{ij}}{1-\theta_{hs}}\right) \tag{B.4.2}$$

$$-\sum_{k=1}^{K}p(c_k)^2\frac{1}{M^2}\cdot\left\{\sum_{j,l}A_2(j,l)\theta_{jk}\theta_{lk}+2\sum_{j}A_1(j)\theta_{jk}+A_0\right\}. \tag{B.4.3}$$

Setting to 0, multiplying by $\theta_{hk}(1-\theta_{hk})$, and arranging terms yields the cubic equation:

$$0 =\frac{2\gamma p(c_k)^2}{M^2}\left(Q\ln|\mathcal{Z}|-\alpha_{|\mathcal{Z}|}\right)A_2(h,h)\theta_{hk}^2 \tag{B.4.4}$$

$$+\frac{2\ln\cdot|\mathcal{Z}|Qp(c_k)}{M^2}\left(\sum_{r,l:(r,l)\neq(k,h)}p(c_r)A_2(h,l)\theta_{lk}+A_1(h)\right)\theta_{hk} \tag{B.4.5}$$

$$-\frac{2\ln\cdot|\mathcal{Z}|\beta_{|\mathcal{Z}|}p(c_k)^2}{M^2}\left(\sum_{l:l\neq h}A_2(h,l)\theta_{lk}+A_1(h)\right)\theta_{hk} \tag{B.4.6}$$

$$-(1-\gamma)\sum_{i=1}^{N}q(c_s|\mathbf{y}_i)\theta_{hk}+(1-\gamma)\sum_{i=1}^{N}q(c_s|\mathbf{y}_i)\mathbf{y}_{ih}. \tag{B.4.7}$$

Since the y_{ij} are binary, we are guaranteed that we obtain a cubic equation. Concavity is assured due to the concavity of the log function in the likelihood term and linearity of the terms from the entropy

bounds. Finally, expanding the product and grouping terms and again using $p_{ij}^b = p(\mathbf{z}_i|y_{\cdot j} = b)$ for presentation:

$$\begin{aligned}
\tilde{\mathcal{L}} = &(1-\gamma)\sum_{k=1}^{K}\sum_{i:z(i)=k}\log p(c_k)\prod_j \theta_{jk}^{\mathbf{y}_{ij}}(1-\theta_{jk})^{1-\mathbf{y}_{ij}} \\
&-\gamma\frac{1}{M^2}\sum_{i=1}^{|\mathcal{D}|}\sum_{k=1}^{K}\Big[p(c_k)\sum_{jl}\left(p_{ij}^1-p_{ij}^0\right)\left(p_{il}^1-p_{il}^0\right)\theta_{jk}+\sum_{jl}\left(p_{il}^0(p_{ij}^1-p_{ij}^0)\right)\theta_{jk} \\
&+\sum_{jl}\left(p_{ij}^0(p_{il}^1-p_{il}^0)\right)\theta_{lk}+\sum_{jl}\left(p_{ij}^0 p_{il}^0\right)\Big]-\gamma\sum_k p(c_k)\log p(c_k) \qquad \text{(B.4.8)} \\
=&(1-\gamma)\sum_{k=1}^{K}\sum_{i:z(i)=k}\log p(c_k)\prod_j \theta_{jk}^{\mathbf{y}_{ij}}(1-\theta_{jk})^{1-\mathbf{y}_{ij}} \\
&-\gamma\frac{1}{M^2}\sum_{c_k}p(c_k)\sum_{jl}A_2(j,l)\cdot\theta_{jk}\theta_{lk}+2\sum_j A_1(j)\cdot\theta_{jk}+A_0-\gamma\sum_k p(c_k)\log p(c_k), \qquad \text{(B.4.9)}
\end{aligned}$$

where we have made use of the property that $A_2(j,l) = A_2(l,j)$.

Appendix C

Proofs for Coordinated Conditional Information Bottleneck (CCIB)

C.1 Derivation of stationary equations

We now derive the stationary conditions for the following functional from (6.1.23):

$$\tilde{F}[p(c|x), q(c), q(y|c), q(y|c,z)] = -\tilde{H}(Y|Z,C) - \rho\tilde{H}(C) + \rho H(C|X) - \lambda\tilde{H}(Y|C) \tag{C.1.1}$$

$$= -\sum_{y,z,c} p(y,z,c)\log q(y|z,c) \tag{C.1.2}$$

$$-\rho\sum_{c} p(c)\log q(c) + \rho\sum_{c,x} p(c|x)p(x)\log p(c|x) \tag{C.1.3}$$

$$-\lambda\sum_{c,y} p(y|c)p(c)\log q(y|c). \tag{C.1.4}$$

$$\tilde{F}[p(c|x), q(c), q(y|c)] = \tilde{H}(C) - H(C|X) + \beta\tilde{H}(Y|C) \tag{C.1.5}$$

$$= -\sum_{c} p(c)\log q(c) + \sum_{c,x} p(c|x)p(x)\log p(c|x) \tag{C.1.6}$$

$$-\beta\sum_{c,y} p(y|c)p(c)\log q(y|c). \tag{C.1.7}$$

We enforce normalization of $q(c)$, $q(y|c)$, and $q(y|c,z)$ by introducing Lagrangian constraints with Lagrange parameters κ for $q(c)$, κ_c for $q(y|c)$, and κ_{cz} for $q(y|c,z)$:

$$\tilde{F}[p(c|x), q(c), q(y|c), q(y|c,z)] = -\sum_{y,c,z} p(y|c,z)p(c,z)\log q(y|c,z) \tag{C.1.8}$$

$$-\rho\sum_{c} p(c)\log q(c) + \rho\sum_{c,x} p(c|x)p(x)\log p(c|x) \tag{C.1.9}$$

$$-\lambda\sum_{c,y} p(y|c)p(c)\log q(y|c) + \kappa\left(1 - \sum_{c} q(c)\right) \tag{C.1.10}$$

$$+\sum_{c}\kappa_c\left(1 - \sum_{y} q(y|c)\right) + \sum_{c,z}\kappa_{c,z}\left(1 - \sum_{y} q(y|c,z)\right). \tag{C.1.11}$$

C.1.1 Optimizing q

We first note that $\tilde{F}$, which we wish to minimize, is convex with respect to $q(c)$, $q(y|c)$, and $q(y|c,z)$ and so is optimized by selecting the critical points for $q(c)$, $q(y|c)$, and $q(y|c,z)$.

Stationary equation for $q(c)$

Then, differentiating with respect to $q(c)$:

$$\frac{\partial\tilde{F}}{\partial q(c)} = -\frac{p(c)}{q(c)} + \kappa. \tag{C.1.12}$$

Setting to 0 and solving:

$$0 = -\frac{p(c)}{q(c)} + \kappa \tag{C.1.13}$$

$$q(c) = \frac{p(c)}{\kappa} = p(c), \tag{C.1.14}$$

where the constraint enforces $-\kappa = \sum_c p(c) = 1$.

Stationary equation for $q(y|c)$

Similarly, for $q(y|c)$

$$\frac{\partial \tilde{F}}{\partial q(y|c)} = -\frac{p(y|c)p(c)}{q(y|c)} + \kappa_c. \tag{C.1.15}$$

Setting to 0 and solving:

$$0 = \frac{p(y|c)p(c)}{q(y|c)} + \kappa_c \tag{C.1.16}$$

$$q(y|c) = \frac{p(y|c)p(c)}{-\kappa_c} = p(y|c), \tag{C.1.17}$$

where the constraint enforces $-\kappa_c = \sum_y p(y|c) = 1$.

Stationary equation for $q(y|c,z)$

And finally, for $q(y|c,z)$

$$\frac{\partial \tilde{F}}{\partial q(y|c,z)} = -\frac{p(y|c,z)p(c,z)}{q(y|c,z)} + \kappa_{cz}. \tag{C.1.18}$$

Setting to 0 and solving:

$$0 = \frac{p(y|c,z)p(c,z)}{q(y|c,z)} + \kappa_c \tag{C.1.19}$$

$$q(y|c,z) = \frac{p(y|c,z)p(c,z)}{-\kappa_c} = p(y|c,z), \tag{C.1.20}$$

where the constraint enforces $-\kappa_{cz} = \sum_y p(y|c,z) = 1$.

C.2 Proof for Section 6.1.2

Given the networks G_{in} and G_{out}, from Section 6.1.2 and using the Multi-Information Bottleneck Principle from [Friedman et al., 2001], the Lagrangian $\mathcal{L}^{(1)}$, would be

$$\mathcal{L}^{(1)} = I(X;C) + I(X;Z) - \beta I(Y;C,Z). \tag{C.2.1}$$

This can be simplified, first using the fact that $I(X;Z)$ is constant and then expanding the information term:

$$\arg\min \mathcal{L}^{(1)} = \arg\min I(X;C) + I(X;Z) - \beta I(Y;C,Z) \tag{C.2.2}$$

$$= \arg\min I(X;C) - \beta \sum p(y,c,z) \log \frac{p(y,c,z)}{p(y)p(c,z)} \tag{C.2.3}$$

$$= \arg\min I(X;C) - \beta \sum p(y,c,z) \log \frac{p(y,c,z)}{p(y)p(c|z)p(z)} \cdot \frac{p(y|z)}{p(y|z)} \tag{C.2.4}$$

$$= \arg\min I(X;C) - \beta \left(I(Y;C|Z) + I(Y;Z) - H(Z) \right) \tag{C.2.5}$$

$$= \arg\min I(X;C) - \beta I(Y;C|Z), \tag{C.2.6}$$

where the last step follows from the fact that $I(Y;Z)$ and $H(Z)$ are constant.

C.3 Proofs for Section 6.3

C.3.1 Derivation of Merger Equations (Lemma 6)

Proof. The derivations are based on those presented in [Slonim and Tishby, 2000] but instead use the identities from (6.3.1):

$$\begin{aligned} p(\bar{c}) =& \sum_{x \in \bar{c}} p(x) p(\bar{c}|x) \\ =& \sum_{x \in \bar{c}} p(x)(p(c_i|x) + p(c_j|x)) \\ =& \sum_{x \in \bar{c}} p(x) p(c_i|x) + \sum_{x \in \bar{c}} p(x) p(c_j|x) \\ =& p(c_i) + p(c_j), \end{aligned} \tag{C.3.1}$$

$$\begin{aligned} p(\bar{c}, z) =& \sum_{x \in \bar{c}} p(x) p(z|x) p(\bar{c}|x) \\ =& \sum_{x \in \bar{c}} p(x) p(z|x)(p(c_i|x) + p(c_j|x)) \\ =& \sum_{x \in \bar{c}} p(x) p(z|x) p(c_i|x) + \sum_{x \in \bar{c}} p(x) p(z|x) p(c_j|x) \\ =& p(c_i, z) + p(c_j, z), \end{aligned} \tag{C.3.2}$$

$$\begin{aligned} p(y,\bar{c},z) =& \sum_{x\in\bar{c}} p(x)p(y|x)p(z|x)p(\bar{c}|x) \\ =& \sum_{x\in\bar{c}} p(x)p(y|x)p(z|x)(p(c_i|x)+p(c_j|x)) \\ =& \sum_{x\in\bar{c}} p(x)p(y|x)p(z|x)p(c_i|x) + \sum_{x\in\bar{c}} p(x)p(y|x)p(z|x)p(c_j|x) \\ =& p(y,c_i,z)+p(y,c_j,z). \end{aligned} \tag{C.3.3}$$

□

C.3.2 Derivation of Merger Cost (Lemma 7)

Proof. Solving for:

$$I(X;C^{aft}) + \beta H(Y|C^{aft},Z) - (I(X;C^{bef}) + \beta H(Y|C^{bef},Z)) \tag{C.3.4}$$

We first consider the conditional entropy terms which differ only by the merged clusters:

$$\begin{aligned} \beta H(Y|C^{aft},Z) - \beta H(Y|C^{bef},Z) =& \beta\left[-\sum_{y,z} p(y,\bar{c},z)\log\frac{p(y,\bar{c},z)}{p(\bar{c},z)}\right. \\ &\left.+\sum_{y,z} p(y,c_i,z)\log\frac{p(y,c_i,z)}{p(c_i,z)} + \sum_{y,z} p(y,c_j,z)\log\frac{p(y,c_j,z)}{p(c_j,z)}\right] && \text{(C.3.5a)} \\ =& \beta\left[-\sum_{y,z} [p(y,c_i,z)+p(y,c_j,z)]\log\frac{p(y,\bar{c},z)}{p(\bar{c},z)}\right. \\ &\left.+\sum_{y,z} p(y,c_i,z)\log\frac{p(y,c_i,z)}{p(c_i,z)} + \sum_{y,z} p(y,c_j,z)\log\frac{p(y,c_j,z)}{p(c_j,z)}\right] && \text{(C.3.5b)} \\ =& \beta\left[-\sum_{y,z} p(y,c_i,z)\log\frac{p(y,\bar{c},z)p(c_i,z)}{p(\bar{c},z)p(y,c_i,z)} - \sum_{y,z} p(y,c_j,z)\log\frac{p(y,\bar{c},z)p(c_j,z)}{p(\bar{c},z)p(y,c_j,z)}\right] && \text{(C.3.5c)} \\ =& \beta\left[\sum_{y,z} p(c_i,z)p(y|c_i,z)\log\frac{p(y|c_i,z)}{p(y|\bar{c},z)} + \sum_{y,z} p(c_j,z)p(y|c_j,z)\log\frac{p(y|c_j,z)}{p(y|\bar{c},z)}\right] && \text{(C.3.5d)} \\ =& \beta\left[\sum_{z} p(c_i,z)KL\left[(p(y|c_i,z)\|p(y|\bar{c},z)\right] + \sum_{z} p(c_j,z)KL\left[(p(y|c_j,z)\|p(y|\bar{c},z)\right]\right] && \text{(C.3.5e)} \\ =& \beta\sum_{z} p(\bar{c},z)\left(\frac{p(c_i,z)}{p(\bar{c},z)}KL\left[(p(y|c_i,z)\|p(y|\bar{c},z)\right] + \frac{p(c_j,z)}{p(\bar{c},z)}KL\left[(p(y|c_j,z)\|p(y|\bar{c},z)\right]\right) && \text{(C.3.5f)} \\ =& \beta\sum_{z} p(\bar{c},z)JS_{\Pi(z)}\left[p(y|c_i,z)\|p(y|c_j,z)\right], && \text{(C.3.5g)} \end{aligned}$$

where (C.3.5b) follows from the self-consistent equation (6.3.7) and where $\Pi(z)$ is defined to be:

$$\Pi(z) = \{\pi_i(z), \pi_j(z)\} = \left\{ \frac{p(c_i, z)}{p(\bar{c}, z)}, \frac{p(c_j, z)}{p(\bar{c}, z)} \right\}, \tag{C.3.6}$$

which satisfies the conditions for the definition of Jensen-Shannon Divergence:

$$0 \leq \pi_i(z), \pi_j(z) \leq 1 \tag{C.3.7a}$$

$$\pi_i(z) + \pi_j(z) = 1 \tag{C.3.7b}$$

$$\pi_i(z)p(y|c_i, z) + \pi_j(z)p(y|c_j, z) = \frac{p(y, c_i, z) + p(y, c_j, z)}{p(\bar{c}, z)} = p(y|\bar{c}, z). \tag{C.3.7c}$$

By similar derivations,

$$\Delta\gamma I(C;Y) = -\gamma I(Y|C^{aft}) + \gamma I(Y|C^{bef}) \tag{C.3.8}$$

$$= p(\bar{c}) JS_{\Pi}\left[p(y|c_i) \| p(y|c_j)\right], \tag{C.3.9}$$

$$\Delta I(C;X) = I(C^{aft};X) - I(C^{bef};X) \tag{C.3.10a}$$

$$= -p(\bar{c}) \cdot JS_{\Pi}\left[p(x|c_i) \| p(x|c_j)\right], \tag{C.3.10b}$$

where

$$\Pi = \{\pi_i, \pi_j\} = \left\{ \frac{p(c_i)}{p(\bar{c})}, \frac{p(c_j)}{p(\bar{c})} \right\}. \tag{C.3.11}$$

Combining (C.3.5g) and (C.3.9) and (C.3.10b) results in the equation for the merger cost of c_i and c_j.

$$\Delta\mathcal{L}(c_i, c_j) = \left[\sum_z p(\bar{c}, z) \cdot JS_{\Pi(z)}\left[p(y|c_i, z) \| p(y|c_j, z)\right] \right] + \gamma p(\bar{c}) JS_{\Pi}\left[p(y|c_i) \| p(y|c_j)\right] - \beta p(\bar{c}) \cdot JS_{\Pi}\left[p(x|c_i) \| p(x|c_j)\right] \tag{C.3.12}$$

In the case of hard assignments, where $p(x|c) = \{0, 1\}$, $H(C|X) = 0$, so the contribution from the

compression term is:

$$\begin{aligned}\Delta I(C;X) &= H(C^{aft}) - H(C^{aft}|X) - (H(C^{bef}) - H(C^{bef}|X)) \\ &= H(C^{aft}) - H(C^{bef}) \\ &= p(c_i)\log\frac{p(\bar{c})}{p(c_i)} + p(c_j)\log\frac{p(\bar{c})}{p(c_j)} \\ &= -p(\bar{c})H(\Pi).\end{aligned} \tag{C.3.13}$$

which yields the merger cost in the case of hard assignments:

$$\Delta\mathcal{L}(c_i,c_j) = -p(\bar{c})H(\Pi) - \beta\sum_z p(\bar{c},z)\cdot JS_{\Pi(z)}\left[p(y|c_i,z)\|p(y|c_j,z)\right]. \tag{C.3.14}$$

□

Appendix D

Proofs for Conditional Ensembles (*CondEns*)

D.1 Proof of Lemma 10

Lemma 10. *There exists a C where:*

$$I(C;X|Z) = \sum_j p(z_j) I(C^j; X|Z = z_j). \tag{D.1.1}$$

Proof. Using the fact that

$$\sum_j p(z_j) I(C^j; X|Z = z_j) = \sum_j p(z_j) \sum_{c \in C^j, x \in X} p(c, x|z_j) \log \frac{p(c|x, z_j)}{p(c|z_j)} \tag{D.1.2}$$

$$= \sum_j p(z_j) \sum_{c \in C^j, x \in X} p(c|x, z_j) p(x|z_j) \log \frac{p(c|x, z_j)}{p(c|z_j)}, \tag{D.1.3}$$

since $p(x|z_j)$ is 1 only for those $j : z(x) = z_j$ and 0 otherwise, the summation over x may be restricted so that:

$$= \sum_j p(z_j) \sum_{c \in C^j} \sum_{x \in z_j} p(c|x, z_j) p(x|z_j) \log \frac{p(c|x, z_j)}{p(c|z_j)} \tag{D.1.4}$$

$$= \sum_j p(z_j) I(\tilde{C}^j; X|Z = z_j), \tag{D.1.5}$$

where in the second step follows because the quantity is summed only over the x in the pre-image set. In fact, this implies

$$\sum_j p(z_j) I(C^j; X|Z = z_j) = \sum_j p(z_j) I(\tilde{C}^j; X|Z = z_j), \tag{D.1.6}$$

which means that in the computation of the expectation is affected only by the assignments $\tilde{C}^j$ per z_j. Thus the other assignments of C^j have no effect and so may take any assignment. In particular they may be set according to their $\tilde{C}^j$.

As $\tilde{C}^j$ are a partition and hence non-overlapping for the x (i.e. each x is given a label by only one $\tilde{C}^j$) so may be combined to produce a single clustering[1] C for all x. This construction thus produces a valid clustering.

Taking such a C yields:

$$\begin{aligned}
\sum_j p(z_j) I(\tilde{C}^j; X|Z = z_j) &= \sum_j p(z_j) I(C^j; X|Z = z_j) && \text{(D.1.7)} \\
&= \sum_j p(z_j) \sum_{c \in C} \sum_{x \in X} p(c|x, z_j) p(x|z_j) \log \frac{p(c|x, z_j)}{p(c|z_j)} && \text{(D.1.8)} \\
&= \sum_j \sum_{c \in C} \sum_{x \in X} p(c, x, z_j) \log \frac{p(c|x, z_j)}{p(c|z_j)} && \text{(D.1.9)} \\
&= I(C; X|Z). && \text{(D.1.10)}
\end{aligned}$$

□

[1] where any mapping between the different labels may be used

Appendix E

Details on Experiments

E.1 Details of the binary synthetic set generation used in Section 8.3.2

We generate m-dimensional binary-valued test sets with two natural independent partitionings, A and B, where A partitions the data into K clusters $A(1), \ldots, A(K)$ and B partitions the data into K clusters $B(1), \ldots, B(K)$. We associate m_A of the features Y with partition A and the remaining $m_B = m - m_A$ with B. Each of the sets of features in $Y(A)$ is further divided into subsets $Y(A_1), \ldots, Y(A_K)$, where the subsets are of equal size: $m_{A(i)} = |Y_{A(i)}| = m_A/K$, and each subset $Y_{A(i)}$ is associated with the specific clusters A_i. Representative profiles $\hat{p}_k \sim \{0,1\}^{m_{A(k)}}$ and $\hat{q}_k \sim \{0,1\}^{m_{Bk}}$ are randomly chosen for all $A(k)$ and $B(k)$. Then, n instances are generated where for each instance membership is randomly chosen from $A \times B$ where each of the $K \times K$ configurations have equal probability. The instance is assigned its feature vector Y from the corresponding $\hat{p}_k$ and $\hat{q}_k$ profiles and noise is added by flipping each feature with probability p_{noise} which defaults to 0.1. The resulting set should contain natural partitionings A and B where the relative strength of the two partitionings can be altered by varying m_A and m_B.

This procedure may be easily extended to an arbitrary number of natural independent partitionings by introducing new sets of representative profiles. This is the approach used for generation of the data sets for successive non-redundant clustering in Chapter 9.