Information-Theoretic Co-Clustering

Inderjit S. Dhillon
inderjit@cs.utexas.edu *

Subramanyam Mallela
manyam@cs.utexas.edu †

Dharmendra S. Modha
dmodha@us.ibm.com ‡

UT CS Technical Report # TR-03-12

April, 2003

Abstract

Two-dimensional contingency or co-occurrence tables arise frequently in important applications such as text, web-log and market-basket data analysis. A basic problem in contingency table analysis is *co-clustering*: *simultaneous clustering* of the rows and columns. A novel theoretical formulation views the contingency table as an empirical joint probability distribution of two discrete random variables and poses the co-clustering problem as an optimization problem in *information theory* — the optimal co-clustering maximizes the mutual information between the clustered random variables subject to constraints on the number of row and column clusters. We present a co-clustering algorithm that monotonically increases the preserved mutual information by intertwining both the row and column clusterings at all stages. Using the practical example of simultaneous word-document clustering, we demonstrate that our algorithm works well in practice, especially in the presence of sparsity.

1 Introduction

Clustering is the grouping together of similar objects [12], and has practical importance in a wide variety of applications such as text, web-log and market-basket data analysis. Typically, the data that arises in these applications is arranged as a contingency or co-occurrence table, such as, word-document co-occurrence table or webpage-user browsing data. Most clustering algorithms focus on one-way clustering, i.e., cluster one dimension of the table based on similarities along the second dimension. For example, documents may be clustered based upon their word distributions while users may be clustered if they visit similar web pages. Clearly, in these applications it is sometimes desirable to *co-cluster* or *simultaneously* cluster both dimensions of the contingency table.

There is a clear duality in co-clustering rows and columns; for example, just as documents can be clustered based upon their word distributions, words may be clustered according to the distribution of their occurrence in documents. A "natural" approach to this problem is to treat the (normalized) non-negative contingency table as a joint probability distribution between two random variables. Information theory can then be used to give a theoretical formulation to the co-clustering problem: the optimal co-clustering is one that leads to the minimum loss in *mutual information* (as shown in Lemma 2.1 this loss is always non-negative).

In this paper, we use this information-theoretic formulation of co-clustering, and after quantifying the loss in mutual information we present a novel algorithm that directly optimizes this loss function.

*Department of Computer Sciences, University of Texas, Austin, TX 78712.
†Department of Computer Sciences, University of Texas, Austin, TX 78712.
‡IBM Almaden Research Center, San Jose, CA.

The resulting algorithm is interesting: it intertwines both row and column clustering at all stages. Row clustering is done by assessing closeness of each row distribution, in relative entropy, to certain "row cluster prototypes". Column clustering is done similarly, and this process is iterated till it converges to a local minimum. Co-clustering differs from ordinary one-sided clustering in that at all stages the row cluster prototypes incorporate column clustering information, and vice versa.

We empirically demonstrate that our co-clustering algorithm alleviates the problems of sparsity and high dimensionality by presenting results on joint word-document clustering. The resulting document clusters are observed to be superior to one-dimensional clustering results and to previously proposed information-theoretic approaches that cluster words before clustering the documents.

A word about notation: upper-case letters such as X, Y, $\hat{X}$, $\hat{Y}$ will denote random variables. Elements of sets will be denoted by lower-case letters such as x and y. Quantities associated with clusters will be "hatted": for example, $\hat{X}$ denotes a random variable obtained from a clustering of X while $\hat{x}$ denotes a cluster. Probability distributions will be denoted by p or q when the random variable is obvious or by $p(X,Y)$, $q(X,Y,\hat{X},\hat{Y})$, $p(Y|x)$, or $q(Y|\hat{x})$ to make the random variable explicit. Logarithms to the base 2 are used throughout.

2 Motivation & Problem Formulation

Let X and Y be discrete random variables that take values in the sets $\{x_1, x_2, \ldots, x_m\}$ and $\{y_1, y_2, \ldots, y_n\}$ respectively. Let $p(X,Y)$ denote the joint probability distribution between X and Y. We will think of $p(X,Y)$ as a $m \times n$ matrix. In practice, if p is not known, it may be estimated using observations. Such a statistical estimate is known as a *two-dimensional contingency table* or as a *two-way frequency table* [9].

We are interested in simultaneously clustering or quantizing X into (at most) k disjoint or hard clusters, and Y into (at most) ℓ disjoint or hard clusters. Let the k clusters of X be written as: $\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k\}$, and let the ℓ clusters of Y be written as: $\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_\ell\}$. In other words, we are interested in finding maps C_X and C_Y,

$$C_X : \{x_1, x_2, \ldots, x_m\} \rightarrow \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k\}$$
$$C_Y : \{y_1, y_2, \ldots, y_n\} \rightarrow \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_\ell\}.$$

For brevity, we will often write $\hat{X} = C_X(X)$ and $\hat{Y} = C_Y(Y)$; $\hat{X}$ and $\hat{Y}$ are random variables that are a deterministic function of X and Y, respectively. Observe that X and Y are clustered separately, that is, $\hat{X}$ is a function of X alone and $\hat{Y}$ is a function of Y alone. But, the partition functions C_X and C_Y are allowed to depend upon the entire joint distribution $p(X,Y)$.

Definition 2.1 *We refer to the tuple (C_X, C_Y) as a co-clustering.*

Suppose we are given a co-clustering. Let us "re-order" the rows of the joint distribution p such that all rows mapping into $\hat{x}_1$ are arranged first, followed by all rows mapping into $\hat{x}_2$, and so on. Similarly, let us "re-order" the columns of the joint distribution p such that all columns mapping into $\hat{y}_1$ are arranged first, followed by all columns mapping into $\hat{y}_2$, and so on. This row-column reordering has the effect of dividing the distribution p into little two-dimensional blocks. We refer to each such block as a *co-cluster*.

A fundamental quantity that measures the amount of information random variable X contains about Y (and vice versa) is the *mutual information* $I(X;Y)$ [3]. We will judge the quality of a co-clustering by the resulting loss in mutual information, $I(X;Y) - I(\hat{X};\hat{Y})$ (any non-trivial co-clustering lowers mutual information, see Lemma 2.1 below).

Definition 2.2 *An* optimal co-clustering *minimizes*

$$I(X;Y) - I(\hat{X};\hat{Y}) \tag{1}$$

subject to the constraints on the number of row and column clusters.

Let us illustrate the situation with an example. Consider the 6×6 matrix below that represents the joint distribution:

$$p(X,Y) = \begin{bmatrix} .05 & .05 & .05 & 0 & 0 & 0 \\ .05 & .05 & .05 & 0 & 0 & 0 \\ 0 & 0 & 0 & .05 & .05 & .05 \\ 0 & 0 & 0 & .05 & .05 & .05 \\ .04 & .04 & 0 & .04 & .04 & .04 \\ .04 & .04 & .04 & 0 & .04 & .04 \end{bmatrix} \tag{2}$$

Looking at the row distributions it is natural to group the rows into three clusters: $\hat{x}_1 = \{x_1, x_2\}$, $\hat{x}_2 = \{x_3, x_4\}$ and $\hat{x}_3 = \{x_5, x_6\}$. Similarly the natural column clustering is: $\hat{y}_1 = \{y_1, y_2, y_3\}$, $\hat{y}_2 = \{y_4, y_5, y_6\}$. The resulting joint distribution $p(\hat{X}, \hat{Y})$, see (6) below, is given by:

$$p(\hat{X}, \hat{Y}) = \begin{bmatrix} .3 & 0 \\ 0 & .3 \\ .2 & .2 \end{bmatrix}. \tag{3}$$

It can be verified that the mutual information lost due to this co-clustering is only .0957, and that any other co-clustering leads to a larger loss in mutual information.

The question is: how can we efficiently search for a co-clustering that minimizes the quantity in (1). The following lemma shows that the loss in mutual information can be expressed as the "distance" of $p(X,Y)$ to an approximation $q(X,Y)$ — this lemma will facilitate our search for the optimal co-clustering.

Lemma 2.1 *For a fixed co-clustering (C_X, C_Y), we can write the loss in mutual information as*

$$I(X;Y) - I(\hat{X};\hat{Y}) = D(p(X,Y)||q(X,Y)), \tag{4}$$

where $D(\cdot||\cdot)$ denotes the Kullback-Leibler divergence, also known as relative entropy, and $q(X,Y)$ is a distribution of the form

$$q(x,y) = p(\hat{x},\hat{y})p(x|\hat{x})p(y|\hat{y}), \tag{5}$$

where $\hat{x} = C_X(x)$ and $\hat{y} = C_Y(y)$.

Proof. Since we are considering hard clustering,

$$\begin{aligned} p(\hat{x},\hat{y}) &= \sum_{x\in\hat{x}}\sum_{y\in\hat{y}} p(x,y), \\ p(\hat{x}) &= \sum_{\hat{y}} p(\hat{x},\hat{y}) = \sum_{x\in\hat{x}} p(x), \\ p(\hat{y}) &= \sum_{\hat{x}} p(\hat{x},\hat{y}) = \sum_{y\in\hat{y}} p(y). \end{aligned} \tag{6}$$

By definition,

$$\begin{aligned} I(X;Y) - I(\hat{X};\hat{Y}) &= \sum_x\sum_y p(x,y)\log\frac{p(x,y)}{p(x)p(y)} - \sum_{\hat{x}}\sum_{\hat{y}} p(\hat{x},\hat{y})\log\frac{p(\hat{x},\hat{y})}{p(\hat{x})p(\hat{y})} \\ &= \sum_{\hat{x}}\sum_{\hat{y}}\sum_{x\in\hat{x}}\sum_{y\in\hat{y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} - \sum_{\hat{x}}\sum_{\hat{y}}\left(\sum_{x\in\hat{x}}\sum_{y\in\hat{y}} p(x,y)\right)\log\frac{p(\hat{x},\hat{y})}{p(\hat{x})p(\hat{y})} \\ &= \sum_{\hat{x}}\sum_{\hat{y}}\sum_{x\in\hat{x}}\sum_{y\in\hat{y}} p(x,y)\log\frac{p(x,y)}{p(\hat{x},\hat{y})\frac{p(x)}{p(\hat{x})}\frac{p(y)}{p(\hat{y})}} \\ &= \sum_{\hat{x}}\sum_{\hat{y}}\sum_{x\in\hat{x}}\sum_{y\in\hat{y}} p(x,y)\log\frac{p(x,y)}{q(x,y)}, \end{aligned}$$

where the last step follows since $p(x|\hat{x}) = \frac{p(x)}{p(\hat{x})}$ for $\hat{x} = C_X(x)$ and 0 otherwise, and similarly for $p(y|\hat{y})$. □

Lemma 2.1 shows that the loss in mutual information must be non-negative, and reveals that finding an optimal co-clustering is equivalent to finding an approximating distribution q of the form (5) that is close to p in Kullback-Leibler divergence subject to the constraints on the number of row and column clusters. In Section 4 we will develop more intuition about the approximation q. The distribution q preserves marginals of p, that is, for $\hat{x} = C_X(x)$ and $\hat{y} = C_Y(y)$,

$$q(x) = \sum_y q(x,y) = \sum_{\hat{y}} \sum_{y \in \hat{y}} p(\hat{x},\hat{y}) p(x|\hat{x}) p(y|\hat{y}) = p(x,\hat{x}) = p(x).$$

Similarly, $q(y) = p(y)$.

Recall the example $p(X,Y)$ in (2) and the "natural" row and column clusterings that led to (3). It is easy to verify that the corresponding approximation $q(X,Y)$ defined in (5) equals

$$q(X,Y) = \begin{bmatrix} .054 & .054 & .042 & 0 & 0 & 0 \\ .054 & .054 & .042 & 0 & 0 & 0 \\ 0 & 0 & 0 & .042 & .054 & .054 \\ 0 & 0 & 0 & .042 & .054 & .054 \\ .036 & .036 & .028 & .028 & .036 & .036 \\ .036 & .036 & .028 & .028 & .036 & .036 \end{bmatrix}, \tag{7}$$

and that $D(p||q) = .0957$. Note that the marginals of p are preserved by q.

We end this section by providing another motivation based on the theory of source coding and transmission. Let us set-up an artificial data compression problem, where we want to transmit X and Y from a source to a destination. Let us insist that this transmission be done in two-stages: (a) first compute $\hat{X} = C_X(X)$ and $\hat{Y} = C_Y(Y)$, and transmit the cluster identifiers $\hat{X}$ and $\hat{Y}$ *jointly*; and (b) *separately* transmit X given that the destination already knows $\hat{X}$ and transmit Y given that the destination already knows $\hat{Y}$. The first step will require, on an average, at least, $H(\hat{X},\hat{Y})$ bits, and, the second step will require, on an average, $H(X|\hat{X}) + H(Y|\hat{Y})$ bits. For every fixed co-clustering, the average number of bits that must be transmitted from the source to the destination is:

$$H(\hat{X},\hat{Y}) + H(X|\hat{X}) + H(Y|\hat{Y}). \tag{8}$$

However, by noting the parallel between (5) and (8), it easy to show that:

$$H(\hat{X},\hat{Y}) + H(X|\hat{X}) + H(Y|\hat{Y}) - H(X,Y) = D(p(X,Y)||q(X,Y)).$$

Thus, finding an optimal co-clustering is equivalent to finding a code that minimizes (8) subject to the constraints on the number of row and column clusters. Observe that (8) contains the cross-term $H(\hat{X},\hat{Y})$ that captures the interaction between the row and column clusters. This underscores the fact that clustering of rows and columns must interact in a "good" co-clustering. A naive algorithm that clusters rows without paying attention to columns and vice versa will miss this critical interaction that is the essence of co-clustering.

3 Related work

Most of the clustering literature has focused on one-sided clustering algorithms, see [12] for a comprehensive survey. There was some early work on co-clustering, such as [11] which was limited to problems of small sizes and used a local greedy splitting procedure to identify hierarchical row and column clusters. More recently [4] used a bipartite graph formulation and a spectral heuristic that uses eigenvectors to co-cluster documents and words; however, a restriction in [4] was that each word

cluster was associated with a document cluster. We do not impose such a restriction in this paper; in fact, see Section 5 for examples of different types of row and column clusters.

Our information-theoretic formulation of preserving mutual information is similar to the Information Bottleneck method [20]. The Information Bottleneck method was introduced for one-sided clustering, say X to $\hat{X}$, and tries to minimize the quantity $I(X, \hat{X})$ in order to gain compression in addition to maximizing the mutual information $I(\hat{X}, Y)$; the overall quantity considered in [20] is $I(X, \hat{X}) - \beta I(\hat{X}, Y)$ where the parameter β reflects the tradeoff between compression and preservation of mutual information. The Information Bottleneck algorithm yields a "soft" clustering of the data using a procedure similar to the deterministic annealing approach of [16]. A greedy agglomerative hard clustering version of the Information Bottleneck algorithm was used in [1, 19] to cluster words in order to reduce feature size for supervised text classification. For this same task, recently [6] proposed a divisive hard clustering algorithm that directly minimizes the loss in mutual information and was found to result in higher classification accuracies than [1, 19]. All these algorithms were proposed for one-sided clustering.

An agglomerative hard clustering version of the Information Bottleneck algorithm was used in [18] to cluster documents after clustering words. The work in [8] extended the above work to repetitively cluster documents and then words. However, both these papers use heuristic procedures and cluster documents and words independently using an agglomerative algorithm. In contrast, in this paper we first quantify the loss in mutual information due to co-clustering and then propose an algorithm that has a strong theoretical foundation and monotonically reduces this loss function, converging to a local minimum.

Recently, when considering a general clustering framework using Bayesian belief networks, [10] proposed an iterative optimization method that amounts to a multivariate generalization of [20], and, once again, uses deterministic annealing. A later paper [17] presented an agglomerative algorithm for the same problem that has advantages over [10] in that "it is simpler, fully deterministic, and non-parametric. There is no need to identify cluster splits which is rather tricky." However, [17] pointed out that their "agglomeration procedures do not scale linearly with the sample size as top down methods do ...". In this paper, we are concerned with a principled, top-down method that scales well. Finally, the results in [17] amount to first finding a word clustering followed by finding a document clustering, whereas we present a procedure that intertwines word and document clusterings and continually improves both until a suitable local minima is found, and, hence, is a true co-clustering procedure.

4 Co-clustering Algorithm

We now describe a novel algorithm that monotonically decreases the objective function (1). To describe the algorithm and related proofs, it will be more convenient to think of the joint distribution of $X, Y, \hat{X}$, and $\hat{Y}$. Let $p(X, Y, \hat{X}, \hat{Y})$ denote this distribution. Observe that without any loss of generality we can write:

$$p(x, y, \hat{x}, \hat{y}) = p(\hat{x}, \hat{y}) p(x, y | \hat{x}, \hat{y}). \tag{9}$$

By Lemma 2.1, for the purpose of co-clustering, we will seek an approximation to $p(X, Y, \hat{X}, \hat{Y})$ using a distribution $q(X, Y, \hat{X}, \hat{Y})$ of the form:

$$q(x, y, \hat{x}, \hat{y}) = p(\hat{x}, \hat{y}) p(x | \hat{x}) p(y | \hat{y}). \tag{10}$$

The reader may want to compare (5) and (10): observe that the latter is defined for all combinations of x, y, $\hat{x}$, and $\hat{y}$. Note that in (10) if $\hat{x} \neq C_X(x)$ or $\hat{y} \neq C_Y(y)$ then $q(x, y, \hat{x}, \hat{y})$ is zero. We will think of $p(X, Y)$ as a two-dimensional marginal of $p(X, Y, \hat{X}, \hat{Y})$ and $q(X, Y)$ as a two-dimensional marginal of $q(X, Y, \hat{X}, \hat{Y})$. Intuitively, by (9) and (10), within the co-cluster denoted by $\hat{X} = \hat{x}$ and $\hat{Y} = \hat{y}$, we seek to approximate the distribution $p(X, Y | \hat{X} = \hat{x}, \hat{Y} = \hat{y})$ by a distribution of the form

$p(X|\hat{X} = \hat{x})p(Y|\hat{Y} = \hat{y})$. The following proposition (which we state without proof) establishes that there is no harm in adopting such a formulation.

Proposition 4.1 *For every fixed* hard *co-clustering,*

$$D(p(X,Y)||q(X,Y)) = D(p(X,Y,\hat{X},\hat{Y})||q(X,Y,\hat{X},\hat{Y})).$$

We first establish a few simple, but useful equalities that highlight properties of q that are desirable in approximating p.

Proposition 4.2 *For a distribution q of the form (10), the following marginals and conditionals are preserved:*

$$q(\hat{x},\hat{y}) = p(\hat{x},\hat{y}),\ q(x,\hat{x}) = p(x,\hat{x})\ \&\ q(y,\hat{y}) = p(y,\hat{y}). \tag{11}$$

Thus,

$$p(x) = q(x),\ p(y) = q(y),\ p(\hat{x}) = q(\hat{x}),\ p(\hat{y}) = q(\hat{y}), \tag{12}$$

$$p(x|\hat{x}) = q(x|\hat{x}),\ p(y|\hat{y}) = q(y|\hat{y}), \tag{13}$$

$$p(\hat{y}|\hat{x}) = q(\hat{y}|\hat{x}),\ p(\hat{x}|\hat{y}) = q(\hat{x}|\hat{y}) \tag{14}$$

for all $x, y, \hat{x}$, and $\hat{y}$. And, furthermore, if $\hat{y} = C_Y(y)$ and $\hat{x} = C_X(x)$, then

$$q(y|\hat{x}) = q(y|\hat{y})q(\hat{y}|\hat{x}), \tag{15}$$

$$q(x,y,\hat{x},\hat{y}) = p(x)q(y|\hat{x}), \tag{16}$$

and, symmetrically,

$$q(x|\hat{y}) = q(x|\hat{x})q(\hat{x}|\hat{y}), \tag{17}$$

$$q(x,y,\hat{x},\hat{y}) = p(y)q(x|\hat{y}).$$

Proof. The equalities of the marginals in (11) are simple to show and will not be proved here for brevity. Equalities (12), (13), and (14) easily follow from (11). Equation (15) follows from

$$q(y|\hat{x}) = q(y,\hat{y}|\hat{x}) = \frac{q(y,\hat{y},\hat{x})}{q(\hat{x})} = q(y|\hat{y},\hat{x})q(\hat{y}|\hat{x}) = q(y|\hat{y})q(\hat{y}|\hat{x}).$$

Equation (16) follows from

$$\begin{aligned} q(x,y,\hat{x},\hat{y}) &= p(\hat{x},\hat{y})p(x|\hat{x})p(y|\hat{y}) \\ &= p(\hat{x})p(x|\hat{x})p(\hat{y}|\hat{x})p(y|\hat{y}) \\ &= p(x,\hat{x})p(\hat{y}|\hat{x})p(y|\hat{y}) \\ &= p(x)q(\hat{y}|\hat{x})q(y|\hat{y}) \\ &= p(x)q(y|\hat{x}), \end{aligned}$$

where the last equality follows from (15). □

Lemma 2.1 quantifies the loss in mutual information upon co-clustering as the Kullback-Leibler divergence of $p(X,Y)$ to $q(X,Y)$. Next, we use the above proposition to prove a lemma that expresses the loss in mutual information in two revealing ways. This lemma will lead to a "natural" algorithm.

Lemma 4.1 *The loss in mutual information can be expressed as (i) a weighted sum of the relative entropies between row distributions $p(Y|x)$ and "row-lumped" distributions $q(Y|\hat{x})$, or as (ii)*

a weighted sum of the relative entropies between column distributions $p(X|y)$ and "column-lumped" distributions $q(X|\hat{y})$, that is,

$$\begin{aligned} D(p(X,Y,\hat{X},\hat{Y})||q(X,Y,\hat{X},\hat{Y})) &= \sum_{\hat{x}} \sum_{x:C_X(x)=\hat{x}} p(x) D(p(Y|x)||q(Y|\hat{x})), \\ &= \sum_{\hat{y}} \sum_{y:C_Y(y)=\hat{x}} p(y) D(p(X|y)||q(X|\hat{y})). \end{aligned}$$

Proof. We will only show the first equality, the second follows similarly.

$$\begin{aligned} D(p(X,Y,\hat{X},\hat{Y})||q(X,Y,\hat{X},\hat{Y})) &= \sum_{\hat{x},\hat{y}} \sum_{x:C_X(x)=\hat{x},y:C_Y(y)=\hat{y}} p(x,y,\hat{x},\hat{y}) \log \frac{p(x,y,\hat{x},\hat{y})}{q(x,y,\hat{x},\hat{y})} \\ &\overset{(a)}{=} \sum_{\hat{x},\hat{y}} \sum_{x:C_X(x)=\hat{x},y:C_Y(y)=\hat{y}} p(x)p(y|x) \log \frac{p(x)p(y|x)}{p(x)q(y|\hat{x})} \\ &= \sum_{\hat{x}} \sum_{x:C_X(x)=\hat{x}} p(x) \sum_{y} p(y|x) \log \frac{p(y|x)}{q(y|\hat{x})}, \end{aligned}$$

where (a) follows since when $\hat{y} = C_Y(y)$, $\hat{x} = C_X(x)$ we have that $p(x,y,\hat{x},\hat{y}) = p(x,y) = p(x)p(y|x)$ and from (16). □

The significance of Lemma 4.1 follows from the fact that it allows us to express the objective function solely in terms of the row-clustering, or in terms of the column-clustering. Furthermore, it allows us to define the distribution $q(Y|\hat{x})$ as a "row-cluster prototype", and similarly, the distribution $q(X|\hat{y})$ as a "column-cluster prototype". With this intuition, we now present the co-clustering algorithm in Figure 1. The algorithm works as follows. It starts with an initial co-clustering $(C_X^{(0)}, C_Y^{(0)})$ and iteratively refines it to obtain a sequence of co-clusterings: $(C_X^{(1)}, C_Y^{(1)})$, $(C_X^{(2)}, C_Y^{(2)})$, ... Associated with a generic co-clustering $(C_X^{(t)}, C_Y^{(t)})$ in the sequence, we compute the distributions $p^{(t)}$ and $q^{(t)}$ as:

$$\begin{aligned} p^{(t)}(x,y,\hat{x},\hat{y}) &= p^{(t)}(\hat{x},\hat{y}) p^{(t)}(x,y|\hat{x},\hat{y}), \\ q^{(t)}(x,y,\hat{x},\hat{y}) &= p^{(t)}(\hat{x},\hat{y}) p^{(t)}(x|\hat{x}) p^{(t)}(y|\hat{y}). \end{aligned}$$

Observe that while as a function of four variables, $p^{(t)}(x,y,\hat{x},\hat{y})$ depends upon the iteration index t, the marginal $p^{(t)}(x,y)$ is, in fact, independent of t. Hence, we will write $p(x)$, $p(y)$, $p(x|y)$, $p(y|x)$, and $p(x,y)$, respectively, instead of $p^{(t)}(x)$, $p^{(t)}(y)$, $p^{(t)}(x|y)$, $p^{(t)}(y|x)$, and $p^{(t)}(x,y)$.

In Step 1, the algorithm starts with an initial co-clustering $(C_X^{(0)}, C_Y^{(0)})$ and computes the required marginals of the resulting approximation $q^{(0)}$ (the choice of starting points is important, and will be discussed in detail later in Section 5). The algorithm then computes the appropriate "row-cluster prototypes" $q^{(0)}(Y|\hat{x})$. While the reader may wish to think of these as "centroids", observe that $q^{(0)}(Y|\hat{x})$ is not a centroid, i.e.,

$$q^{(0)}(Y|\hat{x}) \neq \frac{1}{|\hat{x}|} \sum_{x \in \hat{x}} p(Y|x),$$

where $|\hat{x}|$ denotes the number of rows in cluster $\hat{x}$. Rather, by (15), for every y, we write

$$q^{(t)}(y|\hat{x}) = q^{(t)}(y|\hat{y}) q^{(t)}(\hat{y}|\hat{x}), \tag{18}$$

where $\hat{y} = C_Y(y)$. Note that (18) gives a formula that would have been difficult to guess *a priori* without the help of analysis. In Step 2, the algorithm "re-assigns" each row x to a new row-cluster whose row-cluster prototype $q^{(t)}(Y|\hat{x})$ is closest to $p(Y|x)$ in Kullback-Leibler divergence. In essence, Step 2 defines a new row-clustering. Also, observe that the column clustering is not changed in Step

Algorithm Co_Clustering(p,k,ℓ,$C_X^\dagger$,$C_Y^\dagger$)

Input: The joint probability distribution $p(X,Y)$, k the desired number of row clusters, and ℓ the desired number of column clusters.

Output: The partition functions $C_X^\dagger$ and $C_Y^\dagger$.

1. Initialization: Set $t = 0$. Start with some initial partition functions $C_X^{(0)}$ and $C_Y^{(0)}$. Compute
$$q^{(0)}(\hat{X},\hat{Y}),\ q^{(0)}(X|\hat{X}),\ q^{(0)}(Y|\hat{Y})$$
and the distributions $q^{(0)}(Y|\hat{x})$, $1 \le \hat{x} \le k$ using (18).
2. Compute row clusters: For each row x, find its new cluster index as
$$C_X^{(t+1)}(x) = \text{argmin}_{\hat{x}}\ D\left(p(Y|x)||q^{(t)}(Y|\hat{x})\right),$$
resolving ties arbitrarily. Let $C_Y^{(t+1)} = C_Y^{(t)}$.
3. Compute
$$q^{(t+1)}(\hat{X},\hat{Y}),\ q^{(t+1)}(X|\hat{X}),\ q^{(t+1)}(Y|\hat{Y})$$
and the distributions $q^{(t+1)}(X|\hat{y})$, $1 \le \hat{y} \le \ell$ using (19).
4. Compute column clusters: For each column y, find its new cluster index as
$$C_Y^{(t+2)}(y) = \text{argmin}_{\hat{y}}\ D\left(p(X|y)||q^{(t+1)}(X|\hat{y})\right),$$
resolving ties arbitrarily. Let $C_X^{(t+2)} = C_X^{(t+1)}$.
5. Compute
$$q^{(t+2)}(\hat{X},\hat{Y}),\ q^{(t+2)}(X|\hat{X}),\ q^{(t+2)}(Y|\hat{Y})$$
and distributions $q^{(t+2)}(Y|\hat{x})$, $1 \le \hat{x} \le k$ using (18).
6. Stop and return $C_X^\dagger = C_X^{(t+2)}$ and $C_Y^\dagger = C_Y^{(t+2)}$, if the change in objective function value, that is, $D(p(X,Y)||q^{(t)}(X,Y)) - D(p(X,Y)||q^{(t+2)}(X,Y))$, is "small" (say 10^{-3}); Else set $t = t+2$ and go to step 2.

Figure 1: Information theoretic co-clustering algorithm that simultaneously clusters both the rows and columns

2. In Step 3, using the new row-clustering and the old column clustering, the algorithm recomputes the required marginals of $q^{(t+1)}$. More importantly, the algorithm recomputes the column-cluster prototypes. Once again, these are not ordinary centroids, but rather by using (17), for every x, we write
$$q^{(t+1)}(x|\hat{y}) = q^{(t+1)}(x|\hat{x})q^{(t+1)}(\hat{x}|\hat{y}), \tag{19}$$
where $\hat{x} = C_X(x)$. Now, in Step 4, the algorithm "re-assigns" each column y to a new column-cluster whose column-cluster prototype $q^{(t+1)}(X|\hat{y})$ is closest to $p(X|y)$ in Kullback-Leibler divergence. Step 4 defines a new column-clustering while holding the row-clustering fixed. In Step 5, the algorithm re-computes marginals of $q^{(t+2)}$. The algorithm keeps iterating Steps 2 through 5 until some desired convergence condition is met. The following reassuring theorem, which is our main result, guarantees convergence.

Theorem 4.1 *Algorithm* Co_Clustering *monotonically decreases the objective function given in Lemma 2.1.*

Proof.

$$
\begin{aligned}
&D(p^{(t)}(X,Y,\hat{X},\hat{Y})||q^{(t)}(X,Y,\hat{X},\hat{Y})) \\
&\overset{(a)}{=} \sum_{\hat{x}} \sum_{x:C_X^{(t)}(x)=\hat{x}} p(x) \sum_y p(y|x) \log \frac{p(y|x)}{q^{(t)}(y|\hat{x})} \\
&\overset{(b)}{\geq} \sum_{\hat{x}} \sum_{x:C_X^{(t)}(x)=\hat{x}} p(x) \sum_y p(y|x) \log \frac{p(y|x)}{q^{(t)}\left(y|C_X^{(t+1)}(x)\right)} \\
&\overset{(c)}{=} \sum_{\hat{x}} \sum_{x:C_X^{(t+1)}(x)=\hat{x}} p(x) \sum_{\hat{y}} \sum_{y:C_Y^{(t+1)}(y)=\hat{y}} p(y|x) \log \frac{p(y|x)}{q^{(t)}(\hat{y}|\hat{x})\, q^{(t)}(y|\hat{y})} \\
&= \underbrace{\sum_{\hat{x}} \sum_{x:C_X^{(t+1)}(x)=\hat{x}} p(x) \sum_{\hat{y}} \sum_{y:C_Y^{(t+1)}(y)=\hat{y}} p(y|x) \log \frac{p(y|x)}{q^{(t)}(y|\hat{y})}}_{I_1} \\
&\quad + \sum_{\hat{x}} \sum_{x:C_X^{(t+1)}(x)=\hat{x}} p(x) \sum_{\hat{y}} \sum_{y:C_Y^{(t+1)}(y)=\hat{y}} p(y|x) \log \frac{1}{q^{(t)}(\hat{y}|\hat{x})} \\
&= I_1 + \sum_{\hat{x}} \sum_{\hat{y}} \left(\sum_{x:C_X^{(t+1)}(x)=\hat{x}} \sum_{y:C_Y^{(t+1)}(y)=\hat{y}} p(x)p(y|x) \right) \log \frac{1}{q^{(t)}(\hat{y}|\hat{x})} \\
&\overset{(d)}{=} I_1 + \sum_{\hat{x}} \sum_{\hat{y}} \left(q^{(t+1)}(\hat{x},\hat{y}) \right) \log \frac{1}{q^{(t)}(\hat{y}|\hat{x})} \\
&= I_1 + \sum_{\hat{x}} q^{(t+1)}(\hat{x}) \sum_{\hat{y}} q^{(t+1)}(\hat{y}|\hat{x}) \log \frac{1}{q^{(t)}(\hat{y}|\hat{x})} \\
&\overset{(e)}{\geq} I_1 + \sum_{\hat{x}} q^{(t+1)}(\hat{x}) \sum_{\hat{y}} q^{(t+1)}(\hat{y}|\hat{x}) \log \frac{1}{q^{(t+1)}(\hat{y}|\hat{x})} \\
&= \sum_{\hat{x}} \sum_{x:C_X^{(t+1)}(x)=\hat{x}} p(x) \sum_{\hat{y}} \sum_{y:C_Y^{(t+1)}(y)=\hat{y}} p(y|x) \log \frac{p(y|x)}{q^{(t+1)}(\hat{y}|\hat{x}) q^{(t)}(y|\hat{y})} \\
&\overset{(f)}{=} \sum_{\hat{x}} \sum_{x:C_X^{(t+1)}(x)=\hat{x}} p(x) \sum_{\hat{y}} \sum_{y:C_Y^{(t+1)}(y)=\hat{y}} p(y|x) \log \frac{p(y|x)}{q^{(t+1)}(\hat{y}|\hat{x}) q^{(t+1)}(y|\hat{y})} \\
&\overset{(g)}{=} D(p^{(t+1)}(X,Y,\hat{X},\hat{Y})||q^{(t+1)}(X,Y,\hat{X},\hat{Y})), \qquad (20)
\end{aligned}
$$

where (a) follows from Lemma 4.1; (b) follows from Step 2 of the algorithm; (c) follows by rearranging the sum and from (15); (d) follows from Step 3 of the algorithm; (e) follows by non-negativity of the Kullback-Leibler divergence; and (f) follows since we hold the column clusters fixed, that is, $C_Y^{(t+1)} = C_Y^{(t)}$, and (g) is due to (15) and Lemma 4.1.

By using a virtually identical argument, which we omit here for brevity, and by using properties of Steps 4 and 5, we can show that

$$D(p^{(t+1)}(X,Y,\hat{X},\hat{Y})||q^{(t+1)}(X,Y,\hat{X},\hat{Y})) \geq D(p^{(t+2)}(X,Y,\hat{X},\hat{Y})||q^{(t+2)}(X,Y,\hat{X},\hat{Y})). \qquad (21)$$

By combining (20) and (21), it follows that every iteration of the algorithm never increases the objective function. □

Corollary 4.1 *The algorithm in Figure 1 terminates in a finite number of steps at a cluster assignment that is locally optimal, that is, the loss in mutual information cannot be decreased by either (a) re-assignment of a distribution $p(Y|x)$ or $p(X|y)$ to a different cluster distribution $q(Y|\hat{x})$ or $q(X|\hat{y})$, respectively, or by (b) defining a new distribution for any of the existing clusters.*

Proof. The result follows from Theorem 4.1 and since the number of distinct co-clusterings is finite. □

Remark 4.1 A closer examination of the above proof shows that we established that Steps 2 and 3 together imply (20) and Steps 4 and 5 together imply (21). We show how to generalize the above convergence guarantee to a class of iterative algorithms. In particular, any algorithm that uses an arbitrary concatenations of Steps 2 and 3 with Steps 4 and 5 is guaranteed to monotonically decrease the objective function. For example, consider an algorithm that flips a coin at every iteration and performs Steps 2 and 3 if the coin turns up head, and performs Steps 4 and 5 otherwise. As an another example, consider an algorithm that keeps iterating Steps 2 and 3, until no improvement in the objective function is noticed. Next, it can keep iterating Steps 4 and 5, until no further improvement in the objective function is noticed. Now, it can again iterate Steps 2 and 3, and so on and so forth. Both of these algorithms as well all algorithms in the same spirit are guaranteed to monotonically decrease the objective function. Such algorithmic flexibility can allow computationally efficient exploration of various local minimas when starting from a fixed initial random partition in Step 1.

Remark 4.2 While our algorithm is in the spirit of the k-means algorithm, the precise algorithm itself is quite different from the usual k-means. For example, in our algorithm, the distribution $q^{(t)}(Y|\hat{x})$ serves as a "row cluster prototype". This quantity is different from the naive "centroid" of the cluster $\hat{x}$. Similarly, the column cluster prototype $q^{(t+1)}(X|\hat{y})$ is different from the obvious centroid of the cluster $\hat{y}$. In fact, detailed analysis (as is evident from proof of Theorem 4.1) was necessary to identify these key quantities.

Remark 4.3 In this paper, for simplicity, we have restricted attention to co-clustering for joint distributions of two random variables. However, both our algorithm and our main theorem can be easily extended to co-cluster multi-dimensional joint distributions.

4.1 Illustrative Example

We now illustrate how our algorithm works by showing how it discovers the optimal co-clustering for the example $p(X, Y)$ distribution given in (2) of Section 2. Table 1 shows a typical run of our co-clustering algorithm that starts with a random partition of rows and columns. At each iteration Table 1 shows the steps of Algorithm Co_Clustering, the resulting approximation $q^{(t)}(X, Y)$ to the original distribution and the corresponding compressed distribution $p^{(t)}(\hat{X}, \hat{Y})$. The row and column cluster numbers are shown around the matrix to indicate the clustering at each stage. Notice how our intertwined row and column co-clustering leads to progressively better approximations to the original distribution. At the end of four iterations the algorithm almost accurately reconstructs the original distribution, discovers the natural row and column partitions and recovers the ideal compressed distribution $p(\hat{X}, \hat{Y})$ given in (3). A pleasing property to observe is that at all iterations $q^{(t)}(X, Y)$ preserves the marginals of the original $p(X, Y)$.

5 Experimental Results

This section provides empirical evidence to show the benefits of our co-clustering framework and algorithm. In particular we apply the algorithm to the task of document clustering using word-document co-occurrence data as the joint probability distribution. We show that the co-clustering

	$q^{(t)}(X,Y)$						$p^{(t)}(\hat{X},\hat{Y})$	
	$\hat{y}_1$	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_2$		
$\hat{x}_3$	.029	.029	.019	.022	.024	.024	0.10	0.05
$\hat{x}_1$	.036	.036	.014	.028	.018	.018	0.10	0.20
$\hat{x}_2$	.018	.018	.028	.014	.036	.036	0.30	0.25
$\hat{x}_2$	.018	.018	.028	.014	.036	.036		
$\hat{x}_3$	.039	.039	.025	.030	.032	.032		
$\hat{x}_3$	.039	.039	.025	.030	.032	.032		

$\downarrow$ steps 2 & 3 of Figure 1

	$\hat{y}_1$	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_2$		
$\hat{x}_1$	.036	.036	.014	.028	.018	.018	0.20	0.10
$\hat{x}_1$	.036	.036	.014	.028	.018	.018	0.18	0.32
$\hat{x}_2$	.019	.019	.026	.015	.034	.034	0.12	0.08
$\hat{x}_2$	.019	.019	.026	.015	.034	.034		
$\hat{x}_3$	.043	.043	.022	.033	.028	.028		
$\hat{x}_2$	.025	.025	.035	.020	.046	.046		

$\downarrow$ steps 4 & 5 of Figure 1

	$\hat{y}_1$	$\hat{y}_1$	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_2$	$\hat{y}_2$		
$\hat{x}_1$	.054	.054	.042	0	0	0	0.30	0
$\hat{x}_1$	.054	.054	.042	0	0	0	0.12	0.38
$\hat{x}_2$	.013	.013	.010	.031	.041	.041	0.08	0.12
$\hat{x}_2$	.013	.013	.010	.031	.041	.041		
$\hat{x}_3$	.028	.028	.022	.033	.043	.043		
$\hat{x}_2$	.017	.017	.013	.042	.054	.054		

$\downarrow$ steps 2 & 3 of Figure 1

	$\hat{y}_1$	$\hat{y}_1$	$\hat{y}_1$	$\hat{y}_2$	$\hat{y}_2$	$\hat{y}_2$		
$\hat{x}_1$	.054	.054	.042	0	0	0	0.30	0
$\hat{x}_1$	.054	.054	.042	0	0	0	0	0.30
$\hat{x}_2$	0	0	0	.042	.054	.054	0.20	0.20
$\hat{x}_2$	0	0	0	.042	.054	.054		
$\hat{x}_3$	.036	.036	.028	.028	.036	.036		
$\hat{x}_3$	.036	.036	.028	.028	.036	.036		

Table 1: Algorithm Co_Clustering of Figure 1 gives progressively better clusterings and approximations till the optimal is discovered for the example $p(X,Y)$ given in Section 2.

Dataset	Newsgroups included	#docs per group	Total documents
Binary & Binary_subject	talk.politics.mideast, talk.politics.misc	250	500
Multi5 & Multi5_subject	comp.graphics, rec.motorcycles rec.sports.baseball sci.space.talk.politics.mideast	100	500
Multi10_subject	alt.atheism, comp.sys.mac.hardware misc.forsale, rec.autos rec.sport.hockey, sci.crypt, sci.electronics sci.med, sci.space, talk.politics.gun	50	500

Table 2: Datasets: Each data set consists of documents randomly sampled from the respective news groups in the *NG20* corpus.

approach overcomes sparsity yielding substantially better results than the approach of clustering sparse data along a single dimension. We also show better results as compared to previous algorithms in [18] and [8]. The latter algorithms use a greedy technique to cluster documents after words are clustered using the same greedy approach. For brevity we will use the following notation to denote various algorithms in consideration. We call the Information Bottleneck Double Clustering method in [18] as IB-Double and the Iterative Double Clustering algorithm in [8] as IDC. In addition we use 1D-clustering to denote document clustering without any word clustering i.e, clustering along a single dimension.

5.1 Data Sets

For our experimental results we use various subsets of the 20-Newsgroup data(*NG20*) [13] and the *SMART* collection from Cornell (ftp://ftp.cs.cornell.edu/pub/smart).

The *NG20* data set consists of approximately $20,000$ newsgroup articles collected evenly from 20 different usenet news-groups. This data set has been used for testing several supervised text classification tasks [1, 19, 14, 6] and un-supervised document clustering tasks [18, 8]. Many of the news-groups share similar topics and about 4.5% of the documents are cross posted making the boundaries between some news-groups rather fuzzy. To make our comparison consistent with previous algorithms we reconstructed various subsets of *NG20* used in [18, 8]. We applied the same pre-processing steps as in [18] to all the subsets, i.e., removed stop words, ignored file headers and selected the top 2000 words by mutual information[1]. The specific details of the subsets are given in Table 2.

The *SMART* collection consists of MEDLINE, CISI and CRANFIELD sub-collections. MEDLINE consists of 1033 abstracts from medical journals, CISI consists of 1460 abstracts from information retrieval papers and CRANFIELD consists of 1400 abstracts from aerodynamic systems. After removing stop words and numeric characters we selected the top 2000 words by mutual information as part of our pre-processing. We will refer to this data set as *CLASSIC3*.

5.2 Implementation Details

Bow [15] is a library of C code useful for writing text analysis, language modeling and information retrieval programs. We extended Bow to implement co-clustering and document clustering and used MATLAB to give spy plots of the matrices.

[1]The data sets used in [18] and [8] differ in their pre-processing steps. The latter includes subject lines while the former does not. So we prepared two different data sets one with subject lines and the other without subject lines.

5.3 Evaluation Measures

Validating clustering results is a non-trivial task. The relevance of clustering varies from domain to domain. In the presence of true labels, as in the case of data sets used in our experiments, one can form a confusion matrix to measure the effectiveness of the algorithm. Each entry(i, j) in the confusion matrix represents the number of documents in cluster i that belong to true class j. For an objective evaluation measure we use micro-averaged-precision. The idea is to first associate each cluster with the most dominant class label in that cluster. If our confusion matrix is sufficiently diagonal this effectively calculates the fraction of documents along the diagonal to the total number of documents. For each class c in the data set we define $\alpha(c, \hat{y})$ to be the number of documents correctly assigned to c, $\beta(c, \hat{y})$ to be number of documents incorrectly assigned to c and $\gamma(c, \hat{y})$ to be the number of documents incorrectly not assigned to c. The micro-averaged-precision is defined as

$$P(\hat{y}) = \frac{\sum_c \alpha(c, \hat{y})}{\sum_c (\alpha(c, \hat{y}) + \beta(c, \hat{y}))}$$

and micro-averaged-recall is defined as

$$R(\hat{y}) = \frac{\sum_c \alpha(c, \hat{y})}{\sum_c (\alpha(c, \hat{y}) + \gamma(c, \hat{y}))}$$

Note that for uni-labeled data $P(\hat{y}) = R(\hat{y})$.

5.4 Results and Discussion

First we demonstrate that co-clustering is significantly better than clustering along a single dimension using word-document co-occurrence matrices. In all our experiments since we know the number of true document clusters we can give that as input to our algorithm. For example in the case of Binary data set we ask for 2 document clusters. We vary the number of word clusters over the full range of possibilities and give plots showing how co-clustering behaves with varying number of word clusters. A note about our initialization: we use deterministic initialization of word clusters by choosing initial word cluster distributions to be "maximally" far apart from each other[2], and use a random perturbation of the "mean" document to initialize document clusters [5]. Since this initialization has a random component all our results are averages of five trials unless stated otherwise.

Figure 2 shows two confusion matrices obtained on the *CLASSIC3* data set using algorithms 1D-clustering and co-clustering (with 200 word clusters). Observe that co-clustering extracted the original clusters almost correctly resulting in a micro-averaged-precision of 0.9835 while 1D-clustering led to a micro-averaged-precision of 0.9432.

Co-clustering			1D-clustering		
992	4	8	**944**	9	98
40	**1452**	7	71	**1431**	5
1	4	**1387**	18	20	**1297**

Figure 2: Co-clustering accurately recovers original clusters on the *CLASSIC3* data set.

Figure 3 shows confusion matrices obtained by using co-clustering and document clustering on Binary and Binary_subject data sets. While co-clustering achieves 0.852 and 0.946 micro-averaged precision on these data sets respectively, 1D-clustering yielded only 0.67 and 0.648.

We now show that co-clustering can discover structure in sparse word-document matrices. Figure 4 shows the original word-document matrix and the reordered matrix obtained by arranging rows and columns according to cluster order to reveal the various co-clusters. To simplify the figure