Model-based Overlapping Co-Clustering[*]

M. Mahdi Shafiei[†] Evangelos E. Milios[‡]

Abstract

Co-clustering or simultaneous clustering of rows and columns of two-dimensional data matrices, is a data mining technique with various applications such as text clustering and microarray analysis. Most proposed co-clustering algorithms work on the data matrices with special assumptions and they also assume the existence of a number of mutually exclusive row and column clusters, but it is believed that such an ideal structure rarely exists in real data. In this paper, we propose an overlapping co-clustering model which is able to work with any regular exponential family distribution, and corresponding Bregman divergences, thereby making the model applicable to a wide variety of clustering distance functions. The algorithm using a generative model is able to discover overlapping co-clusters in the input data matrix. The necessary algorithms are provided for this model, and the effectiveness of the method is demonstrated through experiments on subsets of Reuters and MovieLens datasets compared to several other clustering methods.

1 Introduction

Grouping similar objects together is called clustering and has been used in variety of applications such as text, web-log and market-basket analysis. In these applications, data is usually represented as a contingency or co-occurrence table such as the term-document matrix in text analysis. Most effort has been devoted to one-way clustering, i.e., clustering one dimension of the table based on similarities along the second dimension but recently there has been a growing interest in developing algorithms which are able to simultaneously cluster both dimensions of the contingency tables.

Most proposed methods partition the data into non-overlapping areas, where each data item belongs to only one cluster. But in some applications, we are trying to find groupings of data where some data points could belong to more than one cluster and thus overlapping clustering is more appropriate. For example, in text processing, when clustering documents into topic categories, documents may contain multiple relevant topics and an overlapping clustering might be more relevant.

In this paper, we extend an approach to one-way model-based overlapping clustering introduced by [1], hereafter called MOC model to a model for overlapping co-clustering. The original algorithm is able to work with any regular exponential family distribution and corresponding Bregman divergences, thereby making the model applicable for a wide variety of clustering distance functions. These algorithm properties are important in areas where the algorithm is dealing with high-dimensional sparse data. Gaussian models and Euclidean distance are known to perform poorly in these situations. Therefore, these attributes are desirable for our proposed algorithm as well.

In order to show the effectiveness of our algorithm, we present experiments in which we used the proposed algorithm to produce overlapping co-clustering for subsets of Reuters and MovieLens data sets. We compare the clustering results with the results produced by the original algorithm, K-Means and information theoretic co-clustering [4] (hereafter referred to as ITCC) algorithm in order to show that using overlapping co-clustering would give us better results in terms of recall and F-Measure.

A brief word on notation: uppercase letters such as X denote a matrix, whose i^{th} row vector is represented as X_i, j^{th} column vector is represented as X^j, and whose entry in row i and column j is represented as X_i^j as well as X_{ij}. X^T represents the transpose of matrix X.

2 Background

Co-clustering can be applied in every situation where a data matrix A is given in which its elements a_{ij} represent the relation between its rows i and its columns j, and we are looking for subsets of rows with certain coherence properties in a subset of the columns. Almost all interesting variants of co-clustering problem are NP-complete. In the simplest case where we have a binary matrix, a co-cluster corresponds to a bi-clique in the corresponding bipartite graph which is a problem known to be NP-complete [11]. For more general cases where we have a matrix with real values, the complexity is necessarily higher than this simple case, since one can use the solution for solving the more restricted version.

Co-clustering has been used successfully in biological applications (See [11] for a complete survey), information retrieval and text mining, collaborative filter- ing, recommendation systems, target marketing and market research, database research, and data mining. In collaborative filtering, co-clustering can be used for identifying groups of cus-

[*] Supported by grants from the Natural Sciences and Engineering Research Council of Canada, IT Interactive Services Inc., and GINIus Inc.

[†] Faculty of Computer Science, Dalhousie University.

[‡] Faculty of Computer Science, Dalhousie University.

tomers with similar interests toward a group of products. The results can be used for target marketing in recommendation systems [5]. One commonly used example of this application is for data matrices where rows indicate customers and columns represent movies [9].

Other model-based clustering and co-clustering approaches that tried to solve the mentioned problems are presented in [3],[4], [6],[7],[8] and [10].

2.1 Mixture Models for Overlapping Co-Clustering Many algorithms have been proposed for co-clustering [11] but most of them assume the existence of a number of mutually exclusive row and column clusters. Although this can be the first step toward extracting knowledge from contingency tables, it is believed that such an ideal structure rarely exist in real data [10].

Even for one-way clustering, there are few algorithms known as "soft" clustering algorithms which can identify overlapping clusters. They first perform probabilistic "soft" clustering by mixture modeling and then make a hard assignment of each data point to one or more clusters using a threshold on cluster membership probability.

Mixture models has been used to develop soft clustering algorithms and one can use them for developing soft co-clustering algorithms as well. Obviously, we can use mixture models to cluster rows and columns separately in order to produce a "soft" co-clustering. But unfortunately the process is unaware of the correspondence between rows and columns. A good co-clustering algorithm should be able to take advantage of interrelations between rows and columns. This process is also ineffective in detecting homogeneous blocks [12]. Because of these problems and believing that using two models for rows and columns is not parsimonious, a block mixture model has been proposed in [12]. The proposed Block Mixture Model is an extension to the formulation of mixture models for the two-dimensional case. They have also proposed several learning schemes for estimating the parameters of the proposed model.

However, there are two problems with using the traditional mixture model formulation. First, the value of the threshold for which we make a hard assignment of each data point to one or more clusters is difficult to learn from given data. Secondly, one can argue that this is not a natural and true generative model for overlapping clustering. In mixture modeling, according to the underlying assumption, a data point is generated only from one mixture component and the membership probability simply gives the probability of that data point being generated from the corresponding mixture component. However, in overlapping clustering, the model should be able to activate multiple mixture components to generate a data point if it belongs to several clusters.

Recently, a model-based clustering algorithm has been proposed which is able to identify overlapping clusters using a true generative model [1]. Considering all features of this algorithm (hereafter referred to as MOC), we propose an extension to it in order to derive our desired co-clustering algorithm. We review the MOC algorithm in the next subsection and then we propose our extension.

2.2 Model-Based Overlapping Clustering (MOC) We present a brief description of Model-Based Overlapping clustering following the same notation in [1]. Given a set of n data points, we represent them by a $n \times d$ matrix X, such that X_i denotes the i^{th} data point and X_i^j represents its j^{th} feature value. In the MOC, every point X_i has a corresponding k-dimensional boolean membership vector M_i where k is the desired number of clusters. The h^{th} component M_i^h of this membership vector is a binary value indicating whether X_i belongs to the h^{th} cluster. So, multiple 1's encode that the point belongs to several clusters.

In this algorithm, the probability of generating all data points is

$$p(X|\Theta) = p(X|M, A) = \prod_{i,j} p(X_i^j|M_i, A^j)$$

where A is the so-called activity matrix of this model. In the MOC, every element A_h^j is interpreted as the activity of cluster h while generating the j^{th} feature of data. In this model, $\Theta = \{M, A\}$ are the parameters of p and X_i^j's are conditionally independent given M_i and A^j. Furthermore, it is assumed that p can be the density function of any regular exponential family distribution, and also assume that the expectation parameter corresponding to X_i is of the form M_iA, so that $E[X_i] = M_iA$.

Using the above assumptions and the bijection between regular exponential distributions and regular Bregman divergences [2], the conditional density can be represented as:

$$p(X_i^j|M_i, A^j) \propto \exp\{-d_\phi(X_i^j, M_iA^j)\}$$

where d_ϕ is the Bregman divergence corresponding to the chosen exponential density p. The MOC tries to select M and A that maximize the probability of generating all data points:

$$p(X, M, A) = p(M, A)p(A|M, A) =$$
$$p(M)p(A)p(X|M, A) =$$
$$\left(\prod_{i,h} p(M_i^h)\right)\left(\prod_{h,j} p(A_h^j)\right)\left(\prod_{i,j} p(X_i^j|M_i, A^j)\right)$$

Assuming independence of M and A and a uniform distribution for A over a sufficiently large compact set implies that $p(M, A) = p(M)p(A) \propto p(M)$. Then, maximizing the log-likelihood of the joint probability gives

$$\max_{M,A} \log p(X, M, A) =$$
$$\min_{M,A}\left[\sum_{i,j} d_\phi(X_{ij}, (MA)_{ij}) - \sum_{i,h} \log \alpha_{ih}\right]$$

where $\alpha_{ih} = p(M_i^h)$ is the (Bernoulli) prior probability of the i^{th} point having a membership M_{ih} to the h^{th} cluster.

3 Model-Based Overlapping Co-Clustering

Our proposed approach for overlapping co-clustering is inspired from one-way overlapping clustering. We consider two boolean membership vectors, one for each data point and one for each feature. We interpret the activity matrix in the MOC model in a different way that would help us to extend the concept to the co-clustering case. We see each element of activity matrix A_h^j as representing the extent that j^{th} feature would contribute in generating that feature if the corresponding data point belonged to only cluster h. In other words, we consider each data value as generated by partial contributions based on different clusters that the corresponding data point belongs to. The membership vector simply shows which clusters take part in generating a particular feature.

In order to extend the idea to the co-clustering case, we assume that the value of these partial contributions A_h^j (partial contribution of feature j in a data point that belongs to data cluster h) is determined by which categories the corresponding feature belongs to. In other words, each one of the categories that the feature belongs to, has a share in the feature contribution A_h^j. Therefore, we can take an approach for modeling A_h^j similar to the approach taken for modeling X_i^j. We consider a membership vector N_j for each feature indicating the feature clusters that this feature belongs to. We also consider matrix C where C_k^l indicates the activity of feature cluster l while generating a data point that belongs to data cluster k. Using a similar notation as MOC, we would have $E[A_i^T] = N_i C$ or $E[A^j] = CN^j$.

For a concrete example, let us consider the term-document matrix. When a document is about to be built, depending on which clusters it belongs to, the weight of one specific word would be the sum of the contributions of that word from the clusters that the document belongs to. Each word can also be selected from several word clusters and thus each word contribution can be derived from several word clusters that the word belongs to. In co-clustering, we assume that instead of words contributing to form document clusters, word clusters would contribute to form a document cluster. In other words, instead of viewing each document cluster as a collection of words, we view a document cluster as a collection of word clusters, where each word cluster is a collection of related words.

This interpretation leads to a similar approach for modeling word clusters. So, instead of assuming one activation matrix that takes care of each specific feature contribution toward document clusters, we view the contribution of j^{th} feature for generating cluster h as the sum of contributions from several different feature clusters. So, instead of matrix A, we consider CN^T where C is a $k \times l$ matrix and N is a $l \times m$ matrix containing m boolean membership vectors representing whether the corresponding feature belongs to a specific feature cluster.

Our extension to co-clustering can also be reached in another way: According to the MOC model, the activity matrix element A_h^j denotes the activity of cluster h when generating the j^{th} feature of data. Each row of A^T corresponds to a feature which is represented by cluster activities while generating it. Therefore, each feature can be represented as a vector of cluster activity values. In this view, we can cluster rows of A^T (features of X) using MOC again. We would have for each feature a membership vector which we call $N_{m\times l}$ and one activity matrix $C_{l\times k}$ so that the expectation parameter corresponding to A_i^T is of the form $N_i C$, so that $E[A_i^T] = N_i C$ or $E[A^j] = CN^j$.

Based on either of these interpretations, the probability of generating all the data points will be

$$p(X|\Theta) = p(X|M,C,N) = \textstyle\prod_{i,j} p(X_i^j|M_i, C, N^j)$$

In this model, $\Theta = \{M, C, N\}$ are the parameters of p and X_i^j are conditionally independent given M_i, C and N^j. Probability density p can be any regular exponential family distribution, where the expectation parameter corresponding to X_i^j is of the form $M_i C N^j$.

Using these assumptions and the bijection between regular exponential distributions and regular Bregman divergences [2], the conditional density can be represented as

$$p(X_i^j|M_i, A^j, N^j, C) \propto \exp\{-d_\phi(X_i^j, M_i C N^j)\}$$

where d_ϕ is the Bregman divergence corresponding to the chosen exponential density p. For example, if p is the Gaussian density, d_ϕ is the squared Euclidean distance [2]. One can estimate the parameters Θ of the most likely model explaining the data by maximizing the log-likelihood of the observed data. The joint distribution of X, M, N and C is given by:

$$\begin{aligned} p(X,M,C,N) &= p(M,C,N)p(A|M,C,N) \\ &= p(M)p(C)p(N)p(X|M,C,N) \\ &= \left(\textstyle\prod_{i,h} p(M_i^h)\right)\left(\textstyle\prod_{i,j} p(C_i^j)\right)\left(\textstyle\prod_{j,l} p(N_l^j)\right) \\ &\quad \left(\textstyle\prod_{i,j} p(X_i^j|M_i, C_i^j, N^j)\right) \end{aligned}$$

As in MOC, we assume that M, C and N are independent of each other apriori and C is distributed uniformly over a sufficiently large compact set, implying that $p(M,C,N) = p(M)p(C)p(N) \propto p(M)p(N)$. Then, maximizing the log-likelihood of the joint distribution gives

$$\max_{M,C,N} \log p(X,M,C,N) =$$

$$\max_{M,C,N} \left[\sum_{i,h} p(M_i^h) + \sum_{j,l} p(N_l^j) \right.$$

$$\left. - \sum_{i,j} d_\phi(X_i^j, M_i C N^j) \right] =$$

$$\min_{M,C,N}\left[\sum_{i,j} d_\phi(X_{ij},(MCN)_{ij}) - \sum_{i,h}\log\alpha_{ih} - \sum_{j,l}\log\beta_{jl}\right]$$

where $\alpha_{ih} = p(M_i^h)$ is the (Bernoulli) prior probability of the i^{th} point (i.e. document) having a membership M_{ih} to the h^{th} row cluster and $\beta_{jl} = p(N_l^j)$ is the (Bernoulli) prior probability of the j^{th} feature (i.e. word) having a membership N_{jl} to the l^{th} column cluster.

4 Algorithms and Analysis

In this section, we propose and analyze algorithms for estimating the overlapping co-clustering model given an observation matrix X. In particular, from a given observation matrix X, we want to estimate the prior matrices α and β, the membership matrices M and N and finally the activity matrix C so as to maximize the log-likelihood of the observation matrix X, assuming $p(X, M, C, N)$, the joint distribution of (X, M, N, C).

The key idea behind the estimation is similar to most algorithms for co-clustering. Each optimization step consists of two similar sub-steps. In each sub-step, we assume that clustering for one dimension of X is fixed and we optimize the objective function considering the other dimension. In the first sub-step, we assume we have fixed column clusters and we try to find optimal row clustering which optimizes the the objective function. In the second sub-step, we assume row clusters being fixed and we do column clustering.

Using this approach, each of these sub-steps would be reduced to an optimization problem similar to [1] with the difference that in each sub-step, we use information obtained from the other sub-step to enhance the clustering result. The outline of the entire optimization process is showed in Alg. 1. Basically we use a minimization technique that alternates between updating α and β, M and N and C. Because the two sub-steps mentioned are similar, we will focus on the first sub-step: having fixed column clusters and trying to find a row clustering which optimizes the objective function.

4.1 Updating α and β The prior matrices α and β are directly calculated from the current estimate of M and N respectively. If we assume that π_h and μ_h represents the prior probability of any row belonging to row cluster h and the prior probability of any column belonging to column cluster l respectively, then, for a particular row i,

$$\alpha_{ih} = \pi_h^{M_i^h}(1-\pi_h)^{1-M_i^h}$$

and for a particular column j,

$$\beta_{jl} = \mu_l^{N_j^l}(1-\mu_l)^{1-N_j^l}.$$

Algorithm 1 MOCC

```
while Row or Column Clusters changes do
    {Assume having fixed column clustering}
    ▷ Update $\alpha$
    ▷ Update $M$ {$N$ and $C$ is assumed known and we try to optimize $M$}
    ▷ Update $C$ {$M$ and $N$ is assumed known and we try to optimize $C$}
    {Assume having fixed row clustering}
    ▷ Update $\beta$
    ▷ Update $N$ {$M$ and $C$ is assumed known and we try to optimize $N$}
    ▷ Update $C$ {$M$ and $N$ is assumed known and we try to optimize $C$}
end while
```

Since π_h and μ_l are the probabilities of Bernoulli random variables, we use the sample mean of the sufficient statistics for the maximum likelihood estimate. The sufficient statistics for Bernoulli is just the indicator of the event. Thus, the maximum likelihood estimates of the priors π_h and μ_l are

$$\pi_h = \tfrac{1}{n}\textstyle\sum_i \mathbb{1}_{\{M_i^h=1\}}$$

and

$$\mu_l = \tfrac{1}{m}\textstyle\sum_j \mathbb{1}_{\{N_j^l=1\}}$$

where $\mathbb{1}_A$ is the indicator function of A.

4.2 Updating M and N We optimize membership matrix M in the first sub-step where we assume that we have a fixed column clustering and therefore the matrix N is assumed fixed. If we use an alternating minimization technique, for a given X,C and N, the update for M has to minimize

$$\textstyle\sum_{i,j} d_\phi(X_{ij},(MCN)_{ij})$$

This objective function is similar to the objective function minimized in [1]. This is an integer optimization problem and there is no known polynomial time algorithm for solving this problem. We basically use a similar algorithm proposed in [1] with some changes in order to handle the new objective function.

For optimizing N in the second sub-step, we follow the same procedure. Actually if one thinks of the transpose of matrix X, then row clusters for X are column clusters for X^T and vice versa. Therefore, the procedure for optimizing M is directly applicable to the optimization of N if we consider X^T as the input matrix.

Since the optimization for M is an integer optimization problem, one simple approach is considering a real relaxation and allowing M to take values in $[0,1]$. Although the real relaxation approach seems simple, the optimization problem resulting from it is not always simple for all Bregman divergences. In the general case, the relaxed optimization problem may not be convex and has inequality constraints. In order to avoid that, we try to solve the integer optimization problem directly and without doing real relaxation.

The problem here like in [1] is a more general form of the subset sum problem: Given a set of k real numbers $a_1, \ldots, a_k$ and a target number x, find a subset such that the sum over the subset is the closest possible to x. In our problem, we use Bregman divergence to measure closeness and we have multiple target numbers to which we want the sum to be close. Similar to the configuration in [1], the problem can be viewed as finding M_i^* such that

$$\begin{aligned} M_i^* &= \arg \min_{M_i \in \{0,1\}^k} d_\phi(X_i, M_i C N) \\ &= \arg \min_{M_i \in \{0,1\}^k} \sum_{j=1}^{m} d_\phi(X_{ij}, \sum_{h=1}^{k} M_i^h C_h N^j) \end{aligned}$$

Therefore, we have m target numbers $X_{i1}, \ldots, X_{im}$, and for each target number X_{ij}, we should choose the subset from $C_1 N^j, \ldots, C_k N^j$. The total loss is the sum of the individual losses, and the problem is to find a single M_i that minimizes the total loss. Using this observation that each point is more likely to be put in low number of clusters, we modify the algorithm proposed in [1] which we call UpdateM (Algorithm 2). There is no theoretical claim that this algorithm is optimal but empirical evidence presented in [1] and also in section 5.2 suggest that it is an efficient algorithm.

The algorithm UpdateM first turns “on” one cluster and then in a greedy manner, searches for the next best cluster to be turned “on” so as to minimize the loss function. If such a cluster is found, then it would be the second cluster turned “on”. Then, it continues this process with the currently turned “on” clusters. In general, if h clusters are turned “on”, UpdateM considers turning each one of the remaining $(k - h)$ clusters ”on”, one at a time. If, at any step, turning ”on” each one of the remaining $(k-h)$ clusters increases the loss function, the search process is terminated. Otherwise, it picks the best $(h+1)^{th}$ cluster to turn “on”, and repeats the search for the next best on the remaining $(k-h-1)$ clusters.

Ideally, this procedure should use all possible permutations as an order for turning “on” clusters to figure out the lowest loss achieved along that particular permutation, and finally choose the best membership vector among all permutations. Obviously, this approach would be infeasible in practice. Instead, UpdateM starts with k so-called threads, one thread for each one of the k clusters turned “on”. Then, in each thread, it performs the above search procedure for adding the next “on” cluster, till no such clusters are found, or all of them have been turned “on”.

Effectively, UpdateM searches over k permutations, each starting with a different cluster turned “on”. The other entries of the permutation are obtained greedily based on the described search procedure. The algorithm has a worst case running time of $O(k^3)$ and is capable of running with any distance function.

Algorithm 2 UpdateM Algorithm

```
▷ Initialize assignment vector $[m]_{1\times k}$ to all zeros
▷ {Separate search thread for each initial cluster turned ”on”}
for $h = 1$ to $k$ do
    ▷ Turn ”on” only the $h^{th}$ cluster, i.e., set $m(h) = 1, m(i) = 0$, if $i \ = h$
    ▷ Set the $h^{th}$ thread $t_h$ to be ‘active’
    ▷ Compute objective function $\ell_h = d(x, mCN^T)$
    ▷ {Run over all possible sizes ($> 1$) of clusters turned ”on”}
    for $r = 2$ to $k$ do
        if thread $t_h$ is still ‘active’ then
            ▷ Set $\ell_h^{old} = \ell_h$
            ▷ From the rest $(k - r + 1)$ clusters, find best cluster to turn ”on”
            if best cluster to turn ”on” is $p$ then
                ▷ Turn ”on” the $p^{th}$ cluster, i.e., $m(p) = 1$
                ▷ Compute objective function $\ell_h = d(x, mCN^T)$
            end if
            if $\ell_h^{old} \leq \ell_h$ then
                ▷ $\ell_h = \ell_h^{old}$
                ▷ Set the $h^{th}$ thread $t_h$ to be ‘inactive’
            end if
        end if
    end for
end for
▷ Set $m = m_0, \ell = d(x, m_0 CN^T)$
▷ Find the best $m$ over all threads using $\ell_h, h = 1, \ldots, k$
▷ If best $m$ over threads is worse than $m_0$, set $m = m_0$
```

4.3 Updating C The next step is updating the activity matrix C where we don’t have the integer restrictions that we had for M and N. The only constraint that such an update needs to satisfy is that MCN stays in the domain of ϕ. For the squared loss case, since the domain of ϕ is $\mathbb{R}$, the problem of

$$\min_C \|X - MCN\|^2$$

is just the standard least squares problem that can be solved exactly by

$$C = M^\dagger X (N^T)^\dagger,$$

where $M^\dagger$ is the pseudo-inverse of M, and is equal to $(M^T M)^{-1} M^T$ in case $M^T M$ is invertible.

In case of I-divergence or un-normalized relative entropy, the problem $\min_C d_I(X, MCN)$ is similar to the problem studied in [1].

5 Experiments

In this section, we describe the details of our experiments that demonstrate the improved performance of MOCC on two real-world data sets, compared to the MOC model, K-Means and information-theoretic co-clustering (ITCC) algorithm.

5.1 Methodology We use two datasets in our experiments: movie recommendation data, and text documents. We use subsets of original datasets with the characteristic that the points in the subset have natural overlapping grouping as explained later on. Using the full data sets is computationally expensive. Therefore, we use these subsets of data in order to make the datasets computationally reasonable to run experiments. But it should be noted that using smaller datasets doesn't make the task much easier, since clustering a small number of points in a high-dimensional space is still a difficult task. Using these subsets of original datasets similar to the ones used in the MOC paper also allows for a consistent comparison with the MOC method.

$100k$ **Movielens dataset - movie recommendations** This is a publicly available dataset from the movie recommendation system developed at the University of Minnesota [1]. The dataset contains $100,000$ ratings for 1682 movies by 943 users. It has user ratings for every movie in the collection: users give ratings on a scale of $1-5$, with 1 indicating extreme dislike and 5 indicating strong approval. There are 943 users in this dataset, but the mean and median number of users voting on any movie are 59 and 27 respectively. Each user has rated at least 20 movies.

As a result, if each movie in this dataset is represented as a vector of ratings over all the users, the vector is high-dimensional but typically very sparse. Each movie has information about the different genres that this movie belongs to. If each genre is considered as a separate category or cluster, then this dataset also has naturally overlapping clusters. Many movies belong to multiple genres, e.g., Aliens belongs to 3 genre categories: action, horror and science fiction. Like in [1], we created 2 subsets from the Movielens dataset:

- *Mv1*: 679 movies from the 3 genres. Animation, Children's and Comedy;
- *Mv2*: 232 movies from the 3 genres. thriller, action and adventure.

We clustered the movies based on the user recommendations to rediscover genres, based on the belief that similarity in recommendation profiles of movies gives an indication about whether they are in related genres.

Text Data Reuters 21578 is currently a very widely used test collection for text categorization research. The data was originally collected and labelled by Carnegie Group, Inc. and Reuters, Ltd [2]. We created a subset from this dataset in the following way: In order to have overlapping classes in the dataset, we removed all those topics which had less than 100 documents and therefore we also removed all those documents that ended up without a topic. We then had 3149 documents belonging to at least one of the six remaining topics. We used document frequency as a feature selection criterion for decreasing the dimensionality of data to 1000. We removed words which had very high document frequencies until we got dimensionality of 1000. We call this subset *RD*.

We used an experimental methodology similar to the one used to demonstrate the effectiveness of the MOC model. For each dataset, we initialized the MOC clustering algorithm by running k-means clustering, where the Euclidian distance was used as the similarity measure and the number of clusters was set to the number of underlying categories in the dataset. The clustering result of MOC, was then used to initialize the MOCC clustering algorithm. For all methods, the number of row clusters was chosen between 2 and 10. The number of column clusters for MOCC and ITCC were chosen between 2 and 10 in order to investigate its effect on the co-clustering result.

In order to compare clustering results, we use precision, recall, and F-measure calculated over pairs of points, as defined in [1]. For each pair of points that share at least one cluster in the overlapping clustering results, these measures try to estimate whether the prediction of this pair as being in the same cluster was correct with respect to the underlying true categories in the data. Precision is calculated as the fraction of pairs correctly put in the same cluster, recall is the fraction of actual pairs that were identified, and F-measure is the harmonic mean of precision and recall.

$$\text{Precision} = \frac{\text{Number of Correctly Identified Pairs}}{\text{Number of Identified Pairs}}$$

$$\text{Recall} = \frac{\text{Number of Correctly Identified Pairs}}{\text{Number of True Pairs}}$$

$$\text{F-measure} = \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$$

[1] http://www.grouplens.org

[2] http://www.daviddlewis.com/resources/testcollections/reuters21578/

Datasets	MOCC	MOC	K-Means	ITCC
Mv1-E1	**0.63**±0.01	0.60±0.04	0.54±0.01	0.29±0.01
Mv1-E2	**0.63**±0.01	0.51±0.03	0.48±0.02	0.22±0.01
Mv2-E1	**0.53**±0.01	0.49±0.04	0.43±0.01	0.32±0.02
Mv2-E2	**0.52**±0.02	0.43±0.06	0.36±0.02	0.25±0.02
RD-E3	**0.32**±0.01	0.29±0.02	0.31±0.01	0.20±0.01
RD-E4	**0.31**±0.01	0.27±0.02	0.26±0.01	0.13±0.00

Table 2: Comparison of results of MOCC, MOC and K-Means algorithm on all datasets in terms of F-Measure. *Mv1* and *Mv2* are two datasets described in section 5.1, E1, E2, E3 and E4 represents different experiments corresponding to computed number of row and column clusters equal to $(6, 6)$, $(10, 10)$, $(6, 10)$ and $(6, 12)$.

5.2 Results Table 1 presents the results of MOCC versus MOC, K-Means and ITCC algorithms in terms of precision and recall for the datasets described in section 5.1 for selected experiments. Each reported result is an average over ten trials.

Table 2 presents the same results in terms of F-Measure. Table 1 shows that for all domains, even though there is no considerable difference between these methods in terms of precision in most cases, the MOCC model has the best recall by a large margin compared to the other three methods: therefore MOCC consistently outperforms the other two methods in terms of overall F-measure as shown in table 2.

The performance of the four examined methods in terms of recall, precision and F-Measure as the number of movie clusters increases for *Mv1* dataset is shown in Fig. 1. With increasing the number of movie clusters, the precision performance of all four methods increases slowly but the recall performance decreases more significantly. The MOCC method consistently performs better than the three other methods in terms of recall and F-Measure. It is worth noting that for any method and for few clusters, having a good recall is not surprising. Any algorithm looking for a few clusters could have relatively good recall performance. The effectiveness of an algorithm shows itself when the number of clusters is high and the algorithm still has a good recall without significant drop in precision. Results show that for the MOCC algorithm, increasing the number of movie clusters has less impact on recall compared to the other three methods. As the number of movie clusters increases, the gap in performance between MOCC and the other methods in terms of recall and F-Measure grows.

The performance of MOCC improves as the number of user clusters increases, as seen in the curves for two different choices of number of user clusters, 2 and 10, as well as intermediate values not shown. As the number of user clusters increases, the performance of MOCC increases. Furthermore, we see that the performance of MOCC for all choices of number of user clusters is better than the other three methods.

Amongst the four methods, ITCC has slightly better precision in most cases but due to its poor recall, its overall performance in terms of F-Measure is significantly worse than the other methods.

Results for *RD* dataset were similar to movie dataset results. Results show that for the MOCC algorithm, increasing the number of document clusters has less impact on recall compared to the other three methods and the gap in performance between MOCC and the other methods in terms of recall and F-Measure grows. Results also show that the performance of MOCC improves as the number of word clusters increases.

6 Conclusions

This paper has introduced a generative model for overlapping co-clustering, MOCC, based on generalizing the MOC model presented in [1]. It uses a generic alternating minimization algorithm for fitting this model to empirical data. We have presented promising experimental results on real news abstracts and movie data. In particular, we have shown evidence that MOCC produces more accurate overlapping clusters than the MOC model, which are significantly better than non-overlapping clusters based on K-Means and ITCC algorithms.

References

[1] A. Banerjee, C. Krumpelman, S. Basu, R. Mooney, and J. Ghosh. Model based overlapping clustering. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, August 2005.

[2] Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh. Clustering with bregman divergences. In *SIAM International Conference on Data Mining (SDM)*, 2004.

[3] Yizong Cheng and George M. Church. Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 93–103. AAAI Press, 2000.

[4] Inderjit S. Dhillon, Subramanyam Mallela, and Dharmendra S. Modha. Information-theoretic co-clustering. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 89–98, New York, NY, USA, 2003.

[5] W. Gaul and M. Schader. A new algorithm for two-mode clustering. In *Data Analysis and Information Systems*, pages 15–23. Springer, 1996.

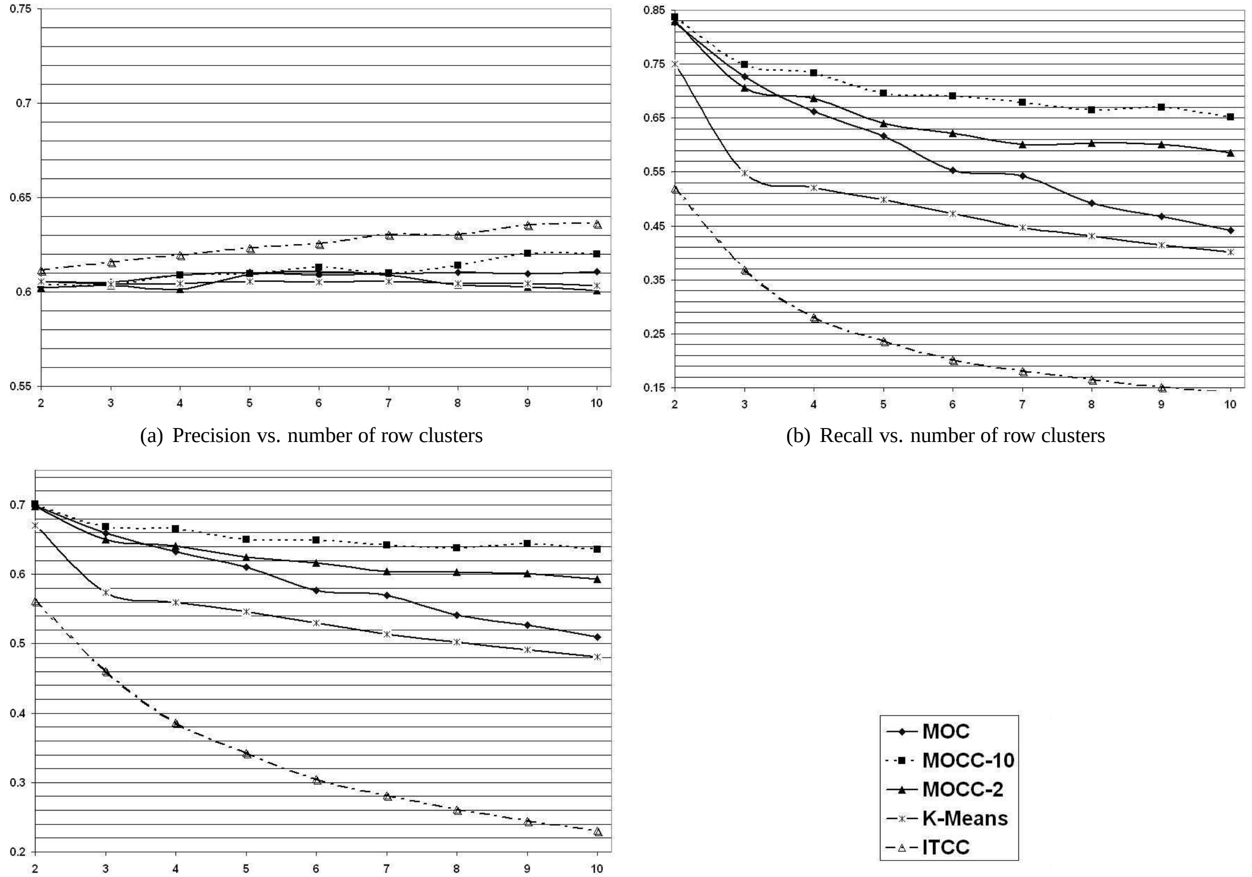
[6] Gérard Govaert and Mohamed Nadif. Clustering with block mixture models. *Pattern Recognition*, 36(2):463–473, 2003.

[7] Thomas Hofmann and Jan Puzicha. Statistical models for co-occurrence data. Technical report, Cambridge, MA, USA, 1998.

[8] Thomas Hofmann and Jan Puzicha. Unsupervised learning from dyadic data. Technical Report TR-98-042, International Computer Science Insitute, Berkeley, CA, 1998.

	Precision				Recall			
Datasets	MOCC	MOC	K-Means	ITCC	MOCC	MOC	K-Means	ITCC
Mv1-E1	0.60±0.01	0.61±0.01	0.60±0.00	**0.63**±0.01	**0.67**±0.02	0.59±0.09	0.48±0.02	0.19±0.01
Mv1-E2	0.62±0.01	0.61±0.01	0.60±0.00	**0.65**±0.01	**0.65**±0.02	0.44±0.05	0.40±0.03	0.13±0.01
Mv2-E1	0.46±0.01	0.47±0.02	0.48±0.01	**0.54**±0.01	**0.62**±0.02	0.51±0.07	0.39±0.01	0.23±0.02
Mv2-E2	0.48±0.01	0.49±0.02	0.49±0.01	**0.57**±0.02	**0.58**±0.04	0.40±0.11	0.29±0.03	0.16±0.02
RD-E3	0.22±0.01	**0.23**±0.01	0.22±0.01	**0.23**±0.00	**0.51**±0.03	0.42±0.08	0.48±0.02	0.18±0.01
RD-E4	0.23±0.00	**0.23**±0.02	0.23±0.00	**0.23**±0.00	**0.48**±0.03	0.33±0.03	0.30±0.03	0.09±0.00

Table 1: Comparison of results of MOCC, MOC and K-Means algorithm on all datasets in terms of precision and recall. *Mv1* and *Mv2* are two datasets described in section 5.1, E1, E2, E3 and E4 represents different experiments corresponding to computed number of row and column clusters equal to $(6, 6)$, $(10, 10)$, $(6, 10)$ and $(6, 12)$.



(a) Precision vs. number of row clusters

(b) Recall vs. number of row clusters

(c) F-Measure vs. number of row clusters

(d) Legend key

Figure 1: Effect of increasing number of row clusters on precision, recall and F-Measure for *Mv1* dataset.

[9] Thomas Hofmann and Jan Puzicha. Latent class models for collaborative filtering. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 688–693, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[10] L. Lazzeroni and A. Owen. Plaid models for gene expression data. *Statistica Sinica*, 12(1):61–86, 2002.

[11] Sara C. Madeira and Arlindo L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM Transaction on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[12] M. Nadif and G. Govaert. Block clustering with mixture model : Comparison between different approaches. In *International Symposium on Applied Stochastic Models and Data Analysis (AMSDA)*, Brest, May 2005.