Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 08-022

Bayesian Co-clustering

Hanhuai Shan and Arindam Banerjee

July 08, 2008

Bayesian Co-clustering

Hanhuai Shan
Dept. of Computer Sci. & Eng.
University of Minnesota, Twin Cities
shan@cs.umn.edu

Arindam Banerjee
Dept. of Computer Sci. & Eng.
University of Minnesota, Twin Cities
banerjee@cs.umn.edu

Abstract

In recent years, co-clustering has emerged as a powerful data mining tool that can analyze dyadic data connecting two entities. However, almost all existing co-clustering techniques are partitional, and allow individual rows and columns of a data matrix to belong to only one cluster. Several current applications, such as recommendation systems and market basket analysis, can substantially benefit from a mixed membership of rows and columns. In this paper, we present Bayesian co-clustering (BCC) models, that allow a mixed membership in row and column clusters. BCC maintains separate Dirichlet priors for rows and columns over the mixed membership and assumes each observation to be generated by an exponential family distribution corresponding to its row and column clusters. We propose a fast variational algorithm for inference and parameter estimation. The model is designed to naturally handle sparse matrices as the inference is done only based on the non-missing entries. In addition to finding co-cluster structure in observations, the model outputs a low dimensional co-embedding, and accurately predicts missing values in the original matrix. We demonstrate the efficacy of the model through experiments on both simulated and real data.

1 Introduction

The application of data mining methods to real life problems has led to an increasing realization that a considerable amount of real data is dyadic, capturing a relation between two entities of interest. For example, *users* rate *movies* in recommendation systems, *customers* purchase *products* in market-basket analysis, etc. Such dyadic data are represented as a matrix with rows and columns representing each entity respectively. An important data mining task pertinent to dyadic data is to get a clustering of each entity, e.g., movie and user groups in recommendation systems, product and consumer groups in market-basket analysis, etc. Traditional clustering algorithms do not perform well on such problems because they are unable to utilize the relationship between the two entities. In comparison, co-clustering [13], i.e., simultaneous clustering of rows and columns of a data matrix, can achieve a much better performance in terms of discovering the structure of data [8] and predicting the missing values [1] by taking advantage of relations between two entities.

Co-clustering has recently received significant attention in algorithm development and applications. [10], [8], and [12] applied co-clustering to text mining, bioinformatics and recommendation systems respectively. [3] proposed a generalized Bregman co-clustering algorithm by considering co-clustering as a matrix approximation problem. While these techniques work reasonably on real data, one important restriction is that almost all of these algorithms are partitional [16], i.e., a row/column belongs to only one row/column cluster. Such an assumption is often restrictive since

objects in real world data typically belong to multiple clusters possibly with varying degrees. For example, a user might be an action movie fan and also a cartoon movie fan. Similar situations arise in most other domains. Therefore, a mixed membership of rows and columns might be more appropriate, and at times essential for describing the structure of such data. It is also expected to substantially benefit the application of co-clustering in such domains.

In this paper, we propose Bayesian co-clustering (BCC) by viewing co-clustering as a generative mixture modeling problem. We assume each row and column to have a mixed membership to row and column clusters respectively. To generate an entry of the data matrix, we first generate the row and column clusters from the corresponding mixed memberships, and then generate the entry given that row-column cluster, i.e., the co-cluster. We introduce separate Dirichlet distributions as Bayesian priors over mixed memberships of rows and columns, effectively averaging the mixture model over all possible mixed memberships. Further, BCC can use any exponential family distribution [4] as the generative model for the co-clusters, which allows the proposed model to be applied to a wide variety of data types, such as real, binary, or discrete matrices. For inference and parameter estimation, we propose an efficient variational EM-style algorithm that preserves dependencies among entries in the same row/column. The model is designed to naturally handle sparse matrices as the inference is done only based on the non-missing entries. Moreover, as a useful by-product, the model accomplishes *co-embedding*, i.e., simultaneous dimensionality reduction of individual rows and columns of the matrix, leading to a simple way to visualize the row/column objects. The efficacy of BCC is demonstrated by the experiments on simulated and real data.

The rest of paper is organized as follows: In Section 2, we present a brief review of generative mixture models. In Section 3, we propose the Bayesian co-clustering model. A variational approach for learning BCC is presented in Section 4. The experimental results are presented in Section 5, and a conclusion is given in Section 6.

2 Generative Mixture Models

In this section, we give a brief overview of existing generative mixture models (GMMs) and co-clustering models based on GMMs as a background for BCC.

Finite Mixture Models. Finite mixture models (FMMs) [9, 4] are one of the most widely studied models for discovering the latent cluster structure from observed data. The density function of a data point $\mathbf{x}$ in FMMs is given by:

$$p(\mathbf{x}|\pi, \Theta) = \sum_{z=1}^{k} p(z|\pi) p(\mathbf{x}|\theta_z) \ ,$$

where π is the prior over k components, and $\Theta = \{\theta_z, [z]_1^k\}$ ($[z]_1^k \equiv z = 1, \ldots, k$) are the parameters of k component distributions $p(\mathbf{x}|\theta_z), [z]_1^k$. In practice, $p(\mathbf{x}|\theta_z)$ could be any exponential family distribution [6] with a density function $p_\psi(\mathbf{x}|\theta) = \exp(\langle \mathbf{x}, \theta \rangle - \psi(\theta))\ p_0(\mathbf{x})$ [4], where θ is the natural parameter, $\psi(\cdot)$ is the cumulant function, and $p_0(\mathbf{x})$ is a non-negative base measure. ψ determines a particular family, such as Gaussian, Poisson, etc., and θ determines a particular distribution in that family. Given a set of observations, the model parameter could be learned by maximum-likelihood estimation using an EM style algorithm [20, 17].

Latent Dirichlet Allocation. One key assumption of a finite mixture model is that the prior π over k components is fixed across all data points. Latent Dirichlet allocation (LDA) [7] relaxes this assumption by assuming there is a separate mixing weight π for each data point, and π is sampled from a Dirichlet distribution $\mathrm{Dir}(\alpha)$. For a sequence of tokens $\mathbf{x} = x_1 \cdots x_d$, LDA with k components has a density of the form

$$p(\mathbf{x}|\alpha, \Theta) = \int_\pi \mathrm{Dir}(\pi|\alpha) \left(\prod_{l=1}^{d} \sum_{z_l=1}^{k} p(z_l|\pi) p(x_l|\theta_{z_l}) \right) d\pi \ .$$

Getting a closed form expression for the marginal density $p(\mathbf{x}|\alpha, \Theta)$ is intractable. Variational inference [7] and Gibbs sampling [11] are two most popular approaches proposed to address the problem.

Bayesian Naive Bayes. While the LDA model relaxes the assumption on the prior, it brings in a limitation on the conditional distribution: LDA can only handle discrete data as tokens. Bayesian naive Bayes (BNB) generalizes LDA to allow the model to work with arbitrarily exponential family distributions [5]. Given a data point $\mathbf{x} = x_1 \cdots x_d$, the density function of the BNB model is as follows:

$$p(\mathbf{x}|\alpha, \Theta, F) = \int_\pi p(\pi|\alpha) \left(\prod_{l=1}^{d} \sum_{z_l=1}^{k} p(z_l|\pi) p_\psi(x_l|z_l, f_l, \Theta) \right) d\pi \ ,$$

where F is the feature set, f_l is the feature for the l^{th} non-missing entry in $\mathbf{x}$, $p_\psi(x_l|z_l, f_l, \Theta)$ could be any exponential family distribution for the component z_l and feature f_l. BNB is able to deal with different types of data, and is designed to handle sparsity.

Co-clustering based on GMMs. The existing literature has a few examples of generative models for co-clustering. Nowicki et al. [19] proposed a stochastic blockstructures model that builds a mixture model for stochastic relationships among objects and identifies the latent cluster via posterior inference. Kemp et al. [14] proposed an infinite relational model that discovers stochastic structure in relational data in form of binary observations. Airoldi et al. [2] recently proposed a mixed membership stochastic blockmodel that relaxes the single-latent-role restriction in stochastic blockstructures model. Such existing models have one or more of the following limitations: (a) The model only handles binary relationships; (b) The model deals with relation within one type of entity, such as a social network among people; (c) There is no computationally efficient algorithm to do inference, and one has to rely on stochastic approximation based on sampling. The proposed BCC model has none of these limitations, and actually goes much further by leveraging the good ideas in such models.

3 Bayesian Co-Clustering

Given an $n_1 \times n_2$ data matrix X, for the purpose of co-clustering, we assume there are k_1 row clusters $\{z_1 = i, [i]_1^{k_1}\}$ and k_2 column clusters $\{z_2 = j, [j]_1^{k_2}\}$. Bayesian co-clustering (BCC) assumes two Dirichlet distributions $\mathrm{Dir}(\alpha_1)$ and $\mathrm{Dir}(\alpha_2)$ for rows and columns respectively, from which the mixing weights π_{1u} and π_{2v} for each row u and each column v are generated. Row clusters for entries in row u and column clusters for entries in column v are sampled from discrete distributions $\mathrm{Disc}(\pi_{1u})$ and $\mathrm{Disc}(\pi_{2v})$ respectively. A row cluster i and a column cluster j together decide a co-cluster (i, j), which has an exponential family distribution $p_\psi(x|\theta_{ij})$, where θ_{ij} is the parameter

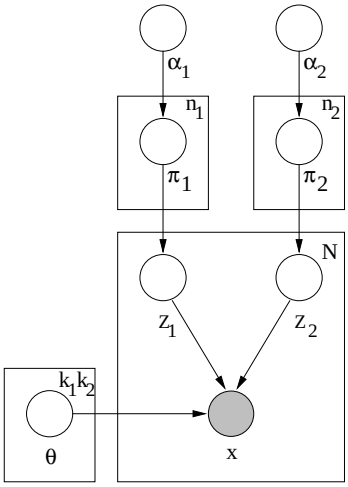


Figure 1: Bayesian co-clustering model.

of the generative model for co-cluster (i, j). For simplicity, we drop ψ from $p_\psi(x|\theta_{ij})$, and the generative process for the whole data matrix is as follows (Figure 1):

1. For each row u,$[u]_1^{n_1}$, choose $\pi_{1u} \sim \text{Dir}(\alpha_1)$.
2. For each column v,$[v]_1^{n2}$, choose $\pi_{2v} \sim \text{Dir}(\alpha_2)$.
3. To generate an entry in row u and column v,
   (a) choose $z_1 \sim \text{Disc}(\pi_{1u})$, $z_2 \sim \text{Disc}(\pi_{2v})$,
   (b) choose $x_{uv} \sim p(x|z_1, z_2, \theta_{z_1 z_2})$.

For this proposed model, the marginal probability of an entry x in the data matrix X is given by:

$$p(x|\alpha_1, \alpha_2, \Theta) = \int_{\pi_1} \int_{\pi_2} p(\pi_1|\alpha_1) p(\pi_2|\alpha_2) \sum_{z_1} \sum_{z_2} p(z_1|\pi_1) p(z_2|\pi_2) p(x|\theta_{z_1 z_2}) d\pi_1 d\pi_2 \ .$$

The probability of the entire matrix is, however, not the product of all such marginal probabilities. That is because π_1 for any row and π_2 for any column are sampled only once for all entries in this row/column. Therefore, the model introduces a coupling between observations in the same row/column, so they are not statistically independent. Note that this is a crucial departure from most mixture models, which assume the joint probability of all data points to be simply a product of the marginal probabilities of each point.

The overall joint distribution over all observable and latent variables is given by

$$\begin{aligned} &p(X, \pi_{1u}, \pi_{2v}, z_{1uv}, z_{2uv}, [u]_1^{n_1}, [v]_1^{n_2} | \alpha_1, \alpha_2, \Theta) \qquad (1) \\ &= \left(\prod_u p(\pi_{1u}|\alpha_1) \right) \left(\prod_v p(\pi_{2v}|\alpha_2) \right) \left(\prod_{u,v} p(z_{1uv}|\pi_{1u}) p(z_{2uv}|\pi_{2v}) p(x_{uv}|\theta_{z_{1uv}, z_{2uv}})^{\delta_{uv}} \right) \ , \end{aligned}$$

where δ_{uv} is an indicator function which takes value 0 when x_{uv} is missing and 1 otherwise, so only the non-missing entries are considered, $z_{1uv} \in \{1, \ldots, k_1\}$ is the latent row cluster and $z_{2uv} \in$

$\{1, \ldots, k_2\}$ is the latent column cluster for observation x_{uv}. Since the observations are conditionally independent given $\{\pi_{1u}, [u]_1^{n_1}\}$ for all rows and $\{\pi_{2v}, [v]_1^{n_2}\}$ for all columns, the joint distribution

$$\begin{aligned} & p(X, \pi_{1u}, \pi_{2v}, [u]_1^{n_1}, [v]_1^{n_2} | \alpha_1, \alpha_2, \Theta) \\ = & \left(\prod_u p(\pi_{1u}|\alpha_1)\right)\left(\prod_v p(\pi_{2v}|\alpha_2)\right)\left(\prod_{u,v} p(x_{uv}|\pi_{1u}, \pi_{2v}, \Theta)^{\delta_{uv}}\right) \ , \end{aligned}$$

where the marginal probability

$$p(x_{uv}|\pi_{1u}, \pi_{2v}, \Theta) = \sum_{z_{1uv}} \sum_{z_{2uv}} p(z_{1uv}|\pi_{1u}) p(z_{2uv}|\pi_{2v}) p(x_{uv}|\theta_{z_{1uv}, z_{2uv}}) \ .$$

Marginalizing over all possible $\{\pi_{1u}, [u]_1^{n_1},\}$ and $\{\pi_{2v}, [v]_1^{n_2}\}$, the probability of observing the entire matrix X is:

$$\begin{aligned} p(X|\alpha_1, \alpha_2, \Theta) = & \int_{\substack{\pi_{1u} \\ u=1,\ldots,n_1}} \int_{\substack{\pi_{2v} \\ v=1,\ldots,n_2}} \left(\prod_u p(\pi_{1u}|\alpha_1)\right)\left(\prod_v p(\pi_{2v}|\alpha_2)\right) \\ & \left(\prod_{u,v} \sum_{z_{1uv}} \sum_{z_{2uv}} p(z_{1uv}|\pi_{1u}) p(z_{2uv}|\pi_{2v}) p(x_{uv}|\theta_{z_{1uv}, z_{2uv}})^{\delta_{uv}}\right) d\pi_{11} \cdots d\pi_{1n_1} d\pi_{21} \cdots d\pi_{2n_2} \ . \end{aligned} \tag{2}$$

It is easy to see (Figure 1) that one-way Bayesian clustering models such as BNB and LDA are special cases of the proposed Baysian co-clustering (BCC) model. Further, BCC inherits all the advantages of BNB and LDA—ability to handle sparsity, applicability to diverse data types using any exponential family distribution, and flexible Bayesian priors with Dirichlet distributions.

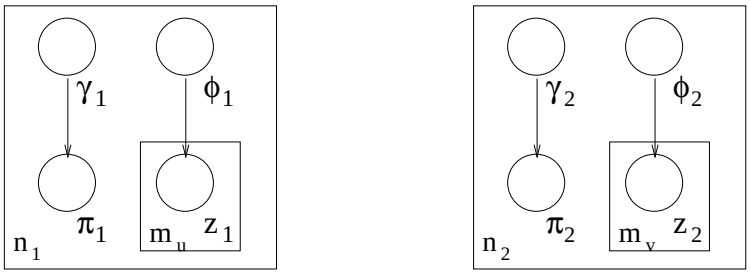
4 Inference and Learning

Given the data matrix X, the learning task for the Bayesian co-clustering model is to estimate the model parameters $(\alpha_1^*, \alpha_2^*, \Theta^*)$ such that the likelihood of observing the matrix X is maximized. A general way is using the expectation maximization (EM) family of algorithms [9]. However, computation of $\log p(X|\alpha_1, \alpha_2, \Theta)$ is intractable for BCC, implying that a direct application of EM is not feasible. In this section, we propose a variational inference method, which alternates between obtaining a tractable lower bound of true log-likelihood $\log p(X|\alpha_1, \alpha_2, \Theta)$, and choosing the model parameters $(\alpha_1, \alpha_2, \Theta)$ to maximize the lower bound. To obtain a tractable lower bound, we consider an entire family of parameterized lower bounds with a set of free variational parameters, and pick the best lower bound from this family by optimizing the lower bound with respect to the free variational parameters.

4.1 Variational Approximation

To get a tractable lower bound for $\log p(X|\alpha_1, \alpha_2, \Theta)$, we introduce $q(\mathbf{z}_1, \mathbf{z}_2, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2 | \boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{\phi}_1, \boldsymbol{\phi}_2)$ as an approximation of the latent variable distribution $p(\mathbf{z}_1, \mathbf{z}_2, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2 | \alpha_1, \alpha_2, \Theta)$:

$$\begin{aligned} & q(\mathbf{z}_1, \mathbf{z}_2, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2 | \boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{\phi}_1, \boldsymbol{\phi}_2) \\ & = \left(\prod_{u=1}^{n_1} q(\pi_{1u}|\gamma_{1u})\right)\left(\prod_{v=1}^{n_2} q(\pi_{2v}|\gamma_{2v})\right)\left(\prod_{u=1}^{n_1}\prod_{v=1}^{n_2} q(z_{1uv}|\phi_{1u}) q(z_{2uv}|\phi_{2v})\right) , \end{aligned}$$



(a) row (b) column

Figure 2: $q(\mathbf{z}_1, \mathbf{z}_2, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2 | \boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{\phi}_1, \boldsymbol{\phi}_2)$ is the variational distribution. $\boldsymbol{\gamma}_1$, $\boldsymbol{\gamma}_2$ are Dirichlet parameters. $\boldsymbol{\phi}_1$, $\boldsymbol{\phi}_2$ are discrete parameters.

where $\boldsymbol{\gamma}_1 = \{\gamma_{1u}, [u]_1^{n_1}\}$ and $\boldsymbol{\gamma}_2 = \{\gamma_{2v}, [v]_1^{n_2}\}$ are variational Dirichlet distribution parameters with k_1 and k_2 dimensions respectively for rows and columns, and $\boldsymbol{\phi}_1 = \{\phi_{1u}, [u]_1^{n_1}\}$ and $\boldsymbol{\phi}_2 = \{\phi_{2v}, [v]_1^{n_2}\}$ are variational discrete distribution parameters with k_1 and k_2 dimensions for rows and columns. Figure 2 shows the approximating distribution $q(\mathbf{z}_1, \mathbf{z}_2, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2 | \boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{\phi}_1, \boldsymbol{\phi}_2)$ as a graphical model, where m_u and m_v are the number of non-missing entries in row u and v. As compared to the variational approximation used in BNB [5] and LDA [7], where the cluster assignment z for every single feature has a variational discrete distribution, in our approximation there is only one variational discrete distribution for an entire row/column. There are at least two advantages of our approach: (a) A single variational discrete distribution for an entire row or column helps maintain the dependencies among all the entries in a row or column; (b) The inference is fast due to the smaller number of variational parameters over which optimization needs to be done.

By a direct application of Jensen's inequality [18], we obtain a lower bound for the true log-likelihood $\log p(X | \alpha_1, \alpha_2, \Theta)$:

$$\begin{aligned} &\log p(X | \alpha_1, \alpha_2, \Theta) \\ &\quad \geq E_q[\log p(X, \mathbf{z}_1, \mathbf{z}_2, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2 | \alpha_1, \alpha_2, \Theta)] - E_q[\log q(\mathbf{z}_1, \mathbf{z}_2, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2 | \boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{\phi}_1, \boldsymbol{\phi}_2)] \ . \end{aligned} \tag{3}$$

We use $L(\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{\phi}_1, \boldsymbol{\phi}_2; \alpha_1, \alpha_2, \Theta)$, or L for brevity, to denote the lower bound. Our algorithm maximizes the family of parameterized lower bounds with respect to the variational parameters $(\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{\phi}_1, \boldsymbol{\phi}_2)$ and the model parameters $(\alpha_1, \alpha_2, \Theta)$ alternately.

4.1.1 Inference

In the inference step, given a choice of model parameters $(\alpha_1, \alpha_2, \Theta)$, the goal is to get the tightest possible lower bound to $\log p(X | \alpha_1, \alpha_2, \Theta)$. It is achieved by maximizing the lower bound $L(\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{\phi}_1, \boldsymbol{\phi}_2; \alpha_1, \alpha_2, \Theta)$ over variational parameters $(\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{\phi}_1, \boldsymbol{\phi}_2)$. While there is no closed form, by taking derivative of L and setting it to zero, the solution can be obtained by iterating over the following set of equations (See Appendix for details):

$$\phi_{1ui} \propto \exp\left(\Psi(\gamma_{1ui}) + \frac{\sum_{v=1}^{n_2} \sum_{j=1}^{k_2} \delta_{uv} \phi_{2vj} \log p(x_{uv} | \theta_{ij})}{m_u}\right) \tag{4}$$

$$\phi_{2vj} \propto \exp\left(\Psi(\gamma_{2vj}) + \frac{\sum_{u=1}^{n_1} \sum_{i=1}^{k_1} \delta_{uv} \phi_{1ui} \log p(x_{uv} | \theta_{ij})}{m_v}\right) \tag{5}$$

$$\gamma_{1ui} = \alpha_{1i} + m_u \phi_{1ui} \tag{6}$$
$$\gamma_{2vj} = \alpha_{2j} + m_v \phi_{2vj} \tag{7}$$

where $[i]_1^{k_1}, [j]_1^{k_2}, [u]_1^{n_1}, [v]_1^{n_2}$, ϕ_{1ui} is the i^{th} component of $\boldsymbol{\phi}_1$ for row u, ϕ_{2vj} is the j^{th} component of $\boldsymbol{\phi}_2$ for column v, and similarly for γ_{1ui} and γ_{2vj}, and $\Psi(\cdot)$ is the digamma function [7]. From a clustering perspective, ϕ_{1ui} denotes the degree of row u belonging to cluster i, for $[u]_1^{n_1}$ and $[i]_1^{k_1}$; and similarly for ϕ_{2vj}.

For our experiments, we use simulated annealing [15] in the inference step to avoid bad local minima. In particular, instead of using (4) and (5) directly for updating ϕ_{1ui} and ϕ_{2vj}, we use

$$\phi_{1ui}^{(t)} \propto (\phi_{1ui})^{1/t}, \quad \phi_{2vj}^{(t)} \propto (\phi_{2vj})^{1/t}$$

at each "temperature" t. At the beginning, $t = \infty$, so the probabilities of row u/column v belonging to all row/column clusters are almost equal. When t slowly decreases step by step, the peak(s) of $\phi_{1ui}^{(t)}$ and $\phi_{2vj}^{(t)}$ gradually show(s) up until we reach $t = 1$, where $\phi_{1ui}^{(1)}$ and $\phi_{2vj}^{(1)}$ become ϕ_{1ui} and ϕ_{2vj}, as in (4) and (5). We then stop decreasing the temperature and keep on updating $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ until convergence. After that, we go on to update $\boldsymbol{\gamma}_1$ and $\boldsymbol{\gamma}_2$ following (6) and (7).

4.1.2 Parameter Estimation

Since choosing parameters to maximize $\log p(X|\alpha_1, \alpha_2, \Theta)$ directly is intractable, we use the optimal lower bound $L(\boldsymbol{\gamma}_1^*, \boldsymbol{\gamma}_2^*, \boldsymbol{\phi}_1^*, \boldsymbol{\phi}_2^*; \alpha_1, \alpha_2, \Theta)$ as the surrogate objective function to be maximized, where $(\boldsymbol{\gamma}_1^*, \boldsymbol{\gamma}_2^*, \boldsymbol{\phi}_1^*, \boldsymbol{\phi}_2^*)$ are the optimum values obtained in the inference step. To estimate the Dirichlet parameters (α_1, α_2), one can use an efficient Newton update as shown in [7, 5] for LDA and BNB. One potential issue with such an update is that an intermediate iterate $\alpha^{(t)}$ can go outside the feasible region $\alpha > 0$. In our implementation, we avoid such a situation using an adaptive line search (See Appendix for details).

For estimating Θ, in principle, a closed form solution is possible for all exponential family distributions. We first consider a special case when the component distributions are univariate Gaussians. The update equations for $\Theta = \{\mu_{ij}, \sigma_{ij}^2, [i]_1^{k_1}, [j]_1^{k_2}\}$ are:

$$\mu_{ij} = \frac{\sum_{u=1}^{n_1} \sum_{v=1}^{n_2} \delta_{uv} x_{uv} \phi_{1ui} \phi_{2vj}}{\sum_{u=1}^{n_1} \sum_{v=1}^{n_2} \delta_{uv} \phi_{1ui} \phi_{2vj}} \tag{8}$$
$$\sigma_{ij}^2 = \frac{\sum_{u=1}^{n_1} \sum_{v=1}^{n_2} \delta_{uv} (x_{uv} - \mu_{ij})^2 \phi_{1ui} \phi_{2vj}}{\sum_{u=1}^{n_1} \sum_{v=1}^{n_2} \delta_{uv} \phi_{1ui} \phi_{2vj}} \ . \tag{9}$$

Following [4], we note that any regular exponential family distribution can be expressed in terms of its expectation parameter m as $p(x|m) = \exp(-d_\phi(x, m))p_0(x)$, where ϕ is the conjugate of the cumulant function ψ of the family and $m = E[X] = \nabla\psi(\theta)$, where θ is the natural parameter. Using the divergence perspective, the estimated mean $M = \{\mu_{ij}, [i]_1^{k_1}, [j]_1^{k_2}\}$ parameter is given by:

$$m_{ij} = \frac{\sum_{u=1}^{n_1} \sum_{v=1}^{n_2} \delta_{uv} x_{uv} \phi_{1ui} \phi_{2vj}}{\sum_{u=1}^{n_1} \sum_{v=1}^{n_2} \delta_{uv} \phi_{1ui} \phi_{2vj}} \ , \tag{10}$$

and $\theta_{ij} = \nabla\phi(m_{ij})$ by conjugacy [4].

4.2 EM Algorithm

Variational inference and parameter estimation lead us to an EM-style alternating maximization algorithm to find the optimal model parameters $(\alpha_1^*, \alpha_2^*, \Theta^*)$ that maximize the lower bound on $\log p(X|\alpha_1, \alpha_2, \Theta)$. In particular, given an initial guess of $(\alpha_1^{(0)}, \alpha_2^{(0)}, \Theta^{(0)})$, the algorithm alternates between two steps till convergence:

1. **E-step**: Given the model parameters $(\alpha_1^{(t)}, \alpha_2^{(t)}, \Theta^{(t)})$, find the variational parameters

$$(\boldsymbol{\gamma}_1^{(t+1)}, \boldsymbol{\gamma}_2^{(t+1)}, \boldsymbol{\phi}_1^{(t+1)}, \boldsymbol{\phi}_2^{(t+1)}) = \underset{(\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{\phi}_1, \boldsymbol{\phi}_2)}{\operatorname{argmax}} L(\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{\phi}_1, \boldsymbol{\phi}_2; \alpha_1^{(t)}, \alpha_2^{(t)}, \Theta^{(t)}) \ .$$

Then, $L(\boldsymbol{\gamma}_1^{(t+1)}, \boldsymbol{\gamma}_2^{(t+1)}, \boldsymbol{\phi}_1^{(t+1)}, \boldsymbol{\phi}_2^{(t+1)}; \alpha_1, \alpha_2, \Theta)$ serves as the lower bound function for the true log-likelihood $\log p(X|\alpha_1, \alpha_2, \Theta)$.

2. **M-step**: An improved estimate of the model parameters can now be obtained as follows:

$$(\alpha_1^{(t+1)}, \alpha_2^{(t+1)}, \Theta^{(t+1)}) = \underset{(\alpha_1, \alpha_2, \Theta)}{\operatorname{argmax}} L(\boldsymbol{\gamma}_1^{(t+1)}, \boldsymbol{\gamma}_2^{(t+1)}, \boldsymbol{\phi}_1^{(t+1)}, \boldsymbol{\phi}_2^{(t+1)}; \alpha_1, \alpha_2, \Theta) \ .$$

5 Experiments

In this section, we present the experimental results on simulated datasets and on real datasets including Movielens, Foodmart, and Jester.

5.1 Simulated Data

Three 80×100 data matrices are generated with 4 row clusters and 5 column clusters, i.e., 20 co-clusters in total, such that each co-cluster generates a 20×20 submatrix. We use one of three exponential family distributions—Gaussian, Bernoulli, and Poisson—as the generative model for each data matrix respectively and each submatrix is generated from the generative model with a predefined parameter, which is set to be different for different submatrices. After generating the data matrix, we randomly permute its rows and columns to yield the final dataset.

For each data matrix, we do semi-supervised initialization by using 5% data in each co-cluster. The results include two parts: parameter estimation and cluster assignment. We compare the estimated parameters with the true model parameters used to generate the data matrix. Further, we evaluate the cluster assignment in terms of cluster accuracy. Cluster accuracy (CA) for rows/columns is defined as: $CA = \frac{1}{n}\sum_{i=1}^{k} nc_i$, where n is the number of rows/columns, k is the number of row/column clusters and nc_i is for the i^{th} row/column result cluster, the largest number of rows/columns that fall into a same true cluster. Since the variational parameters $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ give us the mixing weights for rows and columns, we pick the component with the highest probability as its result cluster.

For all three generative models, we run the algorithm three times and pick the estimated parameters with the highest log-likelihood. Log-likelihood measures the fit of the model to the data, so we are using the model that fits the data best among three runs. Note that no "class label" is used while choosing the model. The comparison of true and estimated parameters after

(a) True (b) Estimated

Figure 3: Parameter estimation for Gaussian.

	Gaussian	**Bernoulli**	**Poisson**
Row	100%	99.5833%	100%
Column	100%	98.5833%	100%

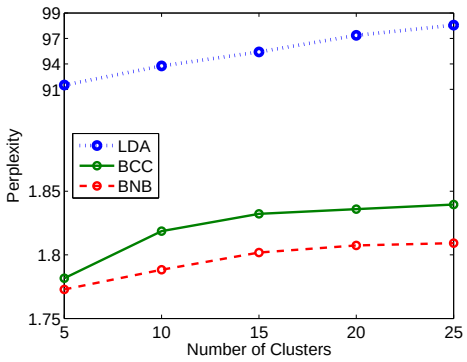
Table 1: Cluster accuracy on simulated data.

alignment for Gaussian case is in Figure 3. The color of each sub-block represents the parameter value for that co-cluster (darker is higher). The result of cluster accuracy is shown in Table 1, which is the average over three runs. From these results, we observe two things: (a) Our algorithm is applicable to different data types by simply choosing an appropriate generative model; (b) We are able to get an accurate parameter estimation and a high cluster accuracy, with semi-supervised initialization by using only 5% of data.

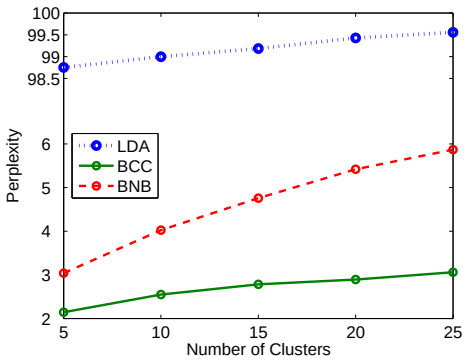
5.2 Real Data

Three real datasets are used in our experiments: (a) **Movieles:**[1] Movielens is a movie recommendation dataset created by the Grouplens Research Project. It contains 100,000 ratings in a sparse data matrix for 1682 movies rated by 943 users. The ratings are ranged from 1 to 5 with 5 the best. We also construct a binarized dataset such that entries whose ratings are higher than 3 become 1 and others become 0. (b) **Jester:**[2] Jester is a joke rating dataset. The original dataset contains 4.1 million continuous ratings of 100 jokes from 73,421 users. The ratings are ranged from -10 to 10 with 10 the best. We pick 1000 users who rate all 100 jokes and use this dense data matrix in our experiment. We also binarize the dataset such that the non-negative entries become 1 and the negative entries become 0. (c) **Foodmart:** Foodmart data comes with Microsoft SQL server. It contains transaction data for a fictitious retailer. In particular, there are 164,558 sales records in a sparse data matrix for 7803 customers and 1559 products. Each record is the number of products bought by the customer. Again, we binarize the dataset such that entries whose number of products are below median are 0 and others are 1. Further, we remove rows and columns with less than 10 non-missing entries. For all three datasets, we use both the binarized and original data in our experiments.

[1] http://www.grouplens.org/node/73
[2] http://goldberg.berkeley.edu/jester-data/

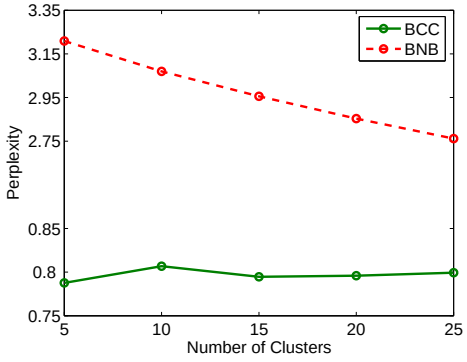


(a) Training Set

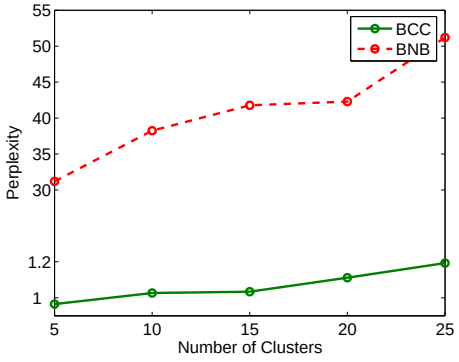


(b) Test Set

Figure 4: Perplexities of BCC, BNB and LDA with varying number of clusters on binarized Jester.



(a) Training Set



(b) Test Set

Figure 5: Perplexities of BCC and BNB with varying number of clusters on original Movielens.

5.2.1 Methodology

For binarized datasets, we use bernoulli distribution as the generative model. For original datasets, we use Discrete, Poisson, and Gaussian as the generative models for Movielens, Foodmart and Jester respectively. For Foodmart data, there is one unit right shift of Poisson distribution since the value of non-missing entries starts from 1 instead of 0, so we substract 1 from all non-missing entries to shift the distribution back.

We train the model from the training set to obtain model parameters. By training, we mean alternating E-step and M-step on training set as described in Section 4 till convergence, so as to obtain model parameters $(\alpha_1^*, \alpha_2^*, \Theta^*)$ that (locally) maximize the variational lower bound on the log-likelihood. We then use the model parameters to do inference, that is, inferring the mixed membership for rows/columns. In particular, there are two steps in our evaluation: (a) Combine training and test data together and do inference (E-step) to obtain variational parameters; (b) Use model parameters and variational parameters to obtain the *perplexity* on the test set. In addition, we also report the perplexity on the training set. Recall that the *perplexity* of a dataset X is defined as [7]:

$$perp(X) = \exp(-\log p(X)/N) \ ,$$

where N is the number of non-missing entries in X. Perplexity monotonically decreases with log-likelihood, implying that *lower perplexity is better* since higher log-likelihood on training set means

	Train set perplexity			Test set perplexity			Test set p-value	
	LDA	BNB	BCC	LDA	BNB	BCC	BCC -LDA	BCC -BNB
Movielens	439.6	**1.70**	1.98	1557.0	3.93	**2.86**	<0.001	<0.001
Foodmart	1461.7	**1.87**	1.95	6542.9	6.48	**2.11**	<0.001	<0.001
Jester	98.3	**1.79**	1.82	98.9	4.02	**2.55**	<0.001	<0.001

(a) On binarized datasets

	Train set perplexity		Test set perplexity		Test set p-value
	BNB	BCC	BNB	BCC	BCC -BNB
Movielens	3.15	**0.81**	38.24	**1.03**	<0.001
Foodmart	4.59	4.59	4.66	**4.60**	<0.001
Jester	**15.46**	18.25	39.94	**24.82**	<0.001

(b) On original datasets

Table 2: Perplexity of BCC, BNB, and LDA on binary and original datasets with 10 clusters. The p-value is obtained from a paired t-test on the differences of test set perplexities between BCC and LDA, as well as between BCC and BNB.

that the model fits the data better, and a higher log-likelihood on the test set implies that the model can explain the data better. For example, in Movielens, a low perplexity on the test set means that the model captures the preference pattern for users such that the model's predicted preferences on test movies for a user would be quite close to his actual preferences; on the contrary, a high perplexity indicates that the user's preference on test movies would be quite different from model's prediction. A similar argument works for Foodmart and Jester as well.

Let X_{train} be the original training set and X_{test} be the original test set. We evaluate the model's prediction performance as follows: We computes variational parameters $(\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{\phi}_1, \boldsymbol{\phi}_2)$ based on (X_{train}, X_{test}), and use them to compute $perp(X_{test})$. We then repeat the process by modifying a fixed percentage of the test set to create $\tilde{X}_{test}$ (noisy data), compute the variational parameters $(\tilde{\boldsymbol{\gamma}}_1, \tilde{\boldsymbol{\gamma}}_2, \tilde{\boldsymbol{\phi}}_1, \tilde{\boldsymbol{\phi}}_2)$ corresponding to $(X_{train}, \tilde{X}_{test})$, and compute $perp(\tilde{X}_{test})$ using these variational parameters. If the model yields a lower perplexity on the true test set than on the modified one, i.e., $perp(X_{test}) < perp(\tilde{X}_{test})$, the model explains X_{test} better than $\tilde{X}_{test}$. If used for prediction based on log-likelihood, the model will accurately predict X_{test}. For a good model, we would expect the perplexity to increase with increasing percentages of test data being modified. Ideally, such an increase will be monotonic, implying that the true test data X_{test} is the most-likely according to the model and a higher perplexity could be used as a sign of more noisy data. In our experiments, since X_{train} is fixed, instead of comparing $perp(X_{test})$ with $perp(\tilde{X}_{test})$ directly, we compare $perp(X_{train}, X_{test})$ with $perp(X_{train}, \tilde{X}_{test})$.

We compare BCC with BNB and LDA in terms of perplexity and prediction performance. Each user/customer is treated as one data point in a row. The comparison with BNB is done on both binarized and original datasets. Similar to BCC, for BNB, we first train the model to estimate model parameters, and then do inference to obtain variational parameters to compute perplexity. The comparison of BCC with LDA is done only on binarized datasets since LDA is not designed to handle real values. To apply LDA, we consider the features with feature value 1 as the tokens appearing in each data point, like the words in a document. For simplicity, we use "row cluster" or "cluster" to refer to the user/customer cluster, and use "column cluster" to refer to the movie, product and joke clusters for BCC on Movielens, Foodmart and Jester datasets respectively. To

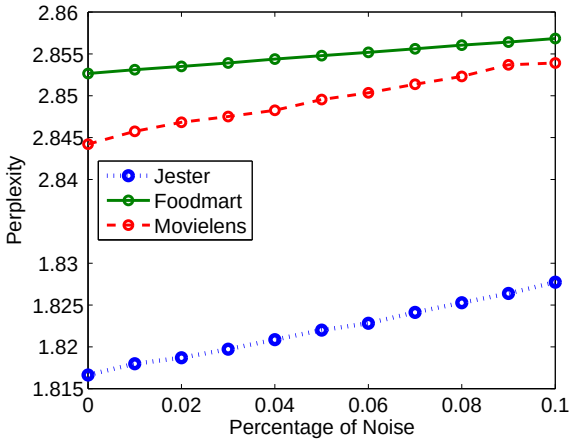


Figure 6: Perplexity curves for Movielens, Foodmart and Jester with increasing noise.

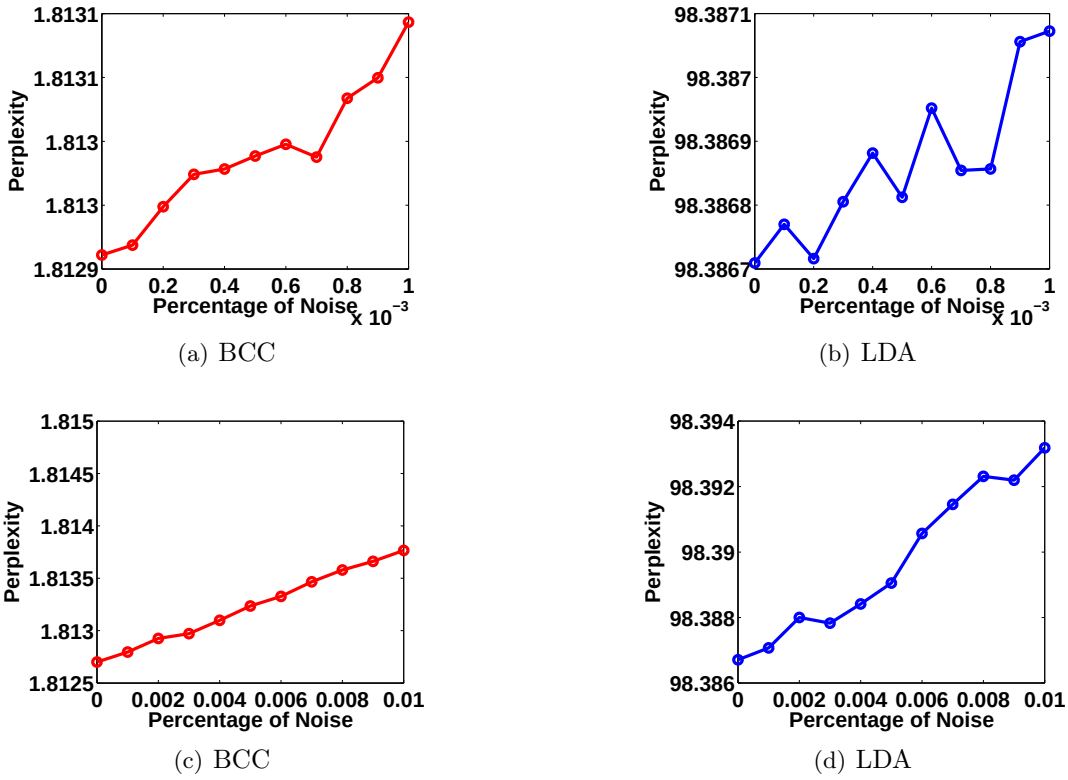
ensure a fair comparison, we do not use simulated annealing for BCC in these experiments.

5.2.2 Results

In this section, we present three main experimental results: (a) Perplexity comparison among BCC, BNB and LDA; (b) The prediction performance comparison between BCC and LDA; (c) The visualization obtained from BCC.

Perplexity Comparison. We compare the perplexity among BCC, BNB and LDA on three datasets with varying number of row clusters from 5 to 25 in steps of 5, and a fixed number of column clusters for BCC to be 20, 10 and 5 for Movielens, Foodmart and Jester respectively. We perform a 10-cross validation with a random initialization, and the results are reported as an average perplexity of 10 runs in Figures 4, 5 and Table 2.

Figure 4 compares the perplexity of BCC, BNB, and LDA on binarized Jester dataset with the number of clusters varying from 5 to 25, and Figure 5 compares the perplexity of BCC and BNB on original Movielens dataset again with varying number of clusters. Note that due to the distinct differences of perplexity among three models, to present the result better, y-axes are not continuous and the unit scales are not all the same. Table 2 presents the perplexities on both binarized and original datasets with fixed 10 row clusters. From these results, there are two observations: (a) For BCC and LDA, the results show that the perplexities of BCC on both training and test sets are 2-3 orders of magnitude lower than that of LDA, and the paired t-test shows that the distinction is statistically significant with an extremely small p-value. The lower perplexity of BCC indicates that BCC fits the data and explains the data better than LDA substantially. (b) For BCC and BNB, although BNB sometimes has a lower perplexity than BCC on training sets, on test sets, the perplexities of BCC are lower than BNB in all cases. Again, the difference is significant based on the paired t-test. BNB's high perplexities on test sets indicate overfitting, especially on the original Movielens data as an example. In comparison, BCC behaves much better than BNB on test sets, possibly because of two reasons: (i) BCC uses much less number of variational parameters than BNB, so as to avoid overfitting; (ii) BCC is able to capture the co-cluster structure which is missing in BNB.



(a) BCC (b) LDA

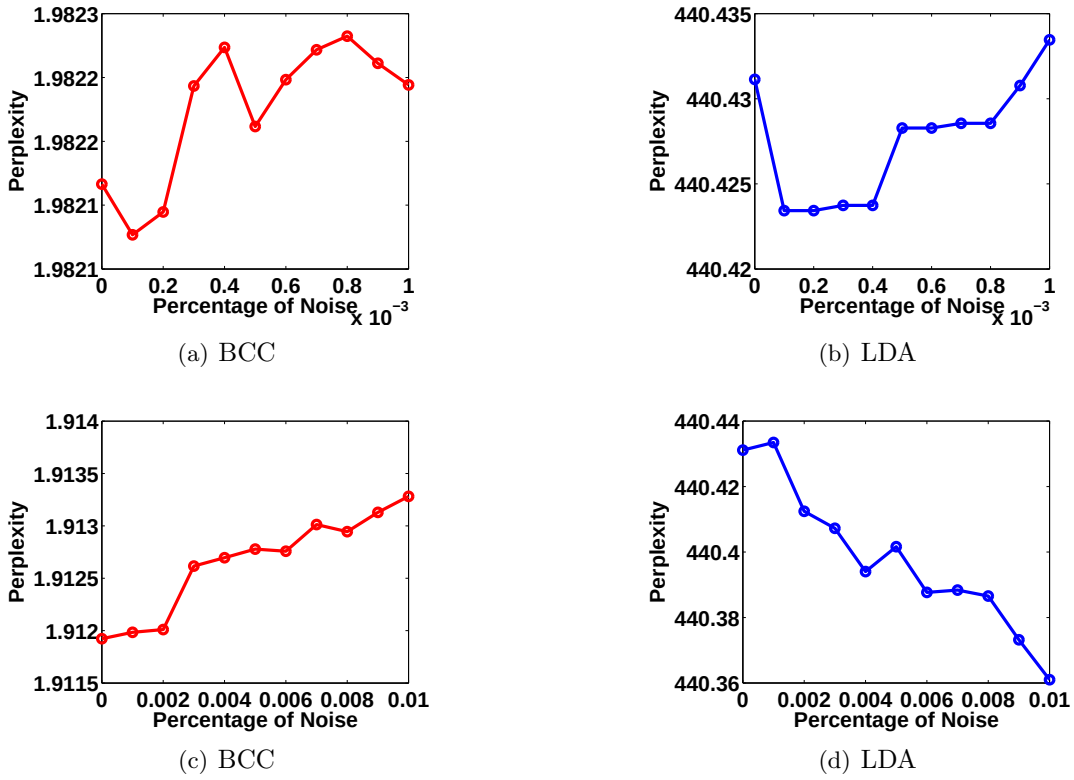
(c) BCC (d) LDA

Figure 7: Perplexity curves of BCC and LDA with increasing noise on binarized Jester.

Prediction Comparison. To evaluate the prediction performance for BCC, we test the perplexity on (X_{train}, X_{test}), as well as on $(X_{train}, \tilde{X}_{test})$ with a certain percentage p of data modified in $\tilde{X}_{test}$. We only compare the prediction on the binarized dataset, which is a reasonable simplification because in real recommendation systems, we usually only need to know whether the user likes (1) the movie/product/joke or not (0), in order to decide whether we should recommend the movie/product/joke. To add noise to binarized data, we flip the entries of 1 to 0 and 0 to 1. We record the perplexities with the percentage of noise p increasing from 1% to 10% in steps of 1% and report the average perplexity of 10 cross validation at each step. The perplexity curves for Movielens, Foodmart and Jester are shown in Figure 6.

At the starting point, with no noise, we have perplexity of data with the true test set X_{test}. At the other extreme end, 10% of the entries in the test set have been modified. As shown in Figure 6, all three lines go up steadily with an increasing percentage of test data modified. This is a surprisingly good result, implying that our model is able to detect increasing noise and convey the message through increasing perplexities. The most accurate result, i.e., the one with the lowest perplexity, is exactly the true test set at the starting point. Therefore, BCC can be used to accurately predict missing values in a matrix.

We add noise at a finer step of modifying 0.1% and 0.01% test data each time, and compare the prediction performance of BCC with LDA. The results on binarized Jester and Movielens datasets are presented in Figure 7 and 8. In both figures, the first row is for adding noise at steps of 0.01% and the second row is for adding noise at steps of 0.1%. The trends of the perplexity curves show the prediction performance. On Jester, we can see that the perplexity curves for BCC in both



(a) BCC (b) LDA

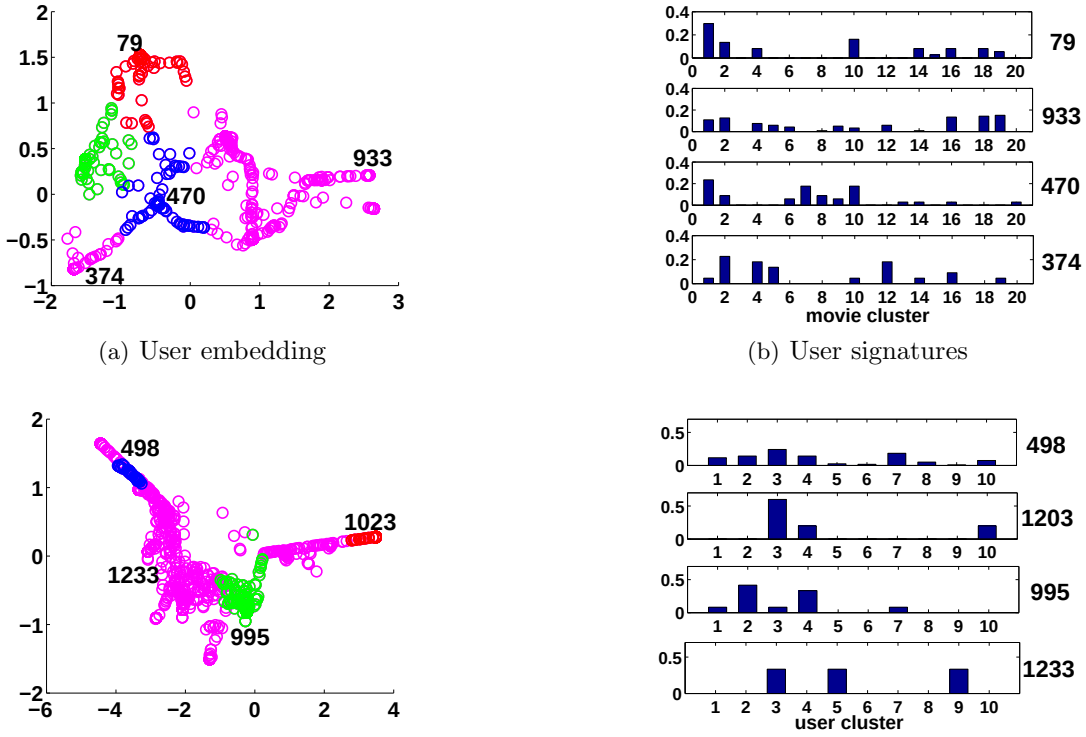
(c) BCC (d) LDA

Figure 8: Perplexity curves of BCC and LDA with increasing noise on binarized Movielens.

Figure 7(a) and 7(c) go up steadily at almost all times. However, the perplexity curves for LDA go up and down from time to time, especially in Figure 7(b), which means that sometimes LDA fits the data with more noise better than that with less noise, indicating a lower prediction accuracy compared with BCC. The difference in prediction performance is even more distinct on the Movielens dataset. When adding noise at steps of 0.01%, there is no clear trend in perplexity curves in Figure 8(a) and 8(b), implying that neither BCC nor LDA is able to detect the noise at this resolution. However, when the step size increases to 0.1%, perplexity curve of BCC starts to go up as in Figure 8(c) but the perplexity curve of LDA goes down as in Figure 8(d). The decreasing perplexity with addition of noise indicates LDA does not have a good prediction performance on Movielens.

Visualization. The co-clustering results give us a compressed representation of the original matrix. We can visualize such compressed matrix to study the relationship between row and column clusters. Figure 10 is an example of user-movie co-clusters on Movielens. There are 10×20 sub-blocks, corresponding to 10 user clusters and 20 movie clusters. The shade for each sub-block shows the user-movie preference. The darker the sub-block is, the more the corresponding movie cluster is preferred by the user cluster. Based on Figure 10, we can see that users in cluster 2 (U2) are a big fan of all kinds of movies, and users in U5 seem uninterested in all movies except those in movie cluster 13 (M13). Moreover, movies in M18 are very popular and preferred by most of the users. In comparison, movies in M4 seems to be far from best sellers. We can also tell that users in U1 prefer M18 the best and M8 the worst. U2 and U6 share several common favorite types of movies.

The variational parameters ϕ_1, with dimension k_1 for rows, and ϕ_2, with dimension k_2 for



(a) User embedding (b) User signatures

(c) Movie embedding (d) Movie signatures

Figure 9: Co-embedding and signatures for users (ϕ_1) and movies (ϕ_2) on Movielens dataset.

columns, give a low-dimensional representation for all the row and column objects. They can be considered as the result of a simultaneous dimensionality reduction over row and column feature vectors. We call the low-dimensional vectors $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ a "co-embedding" since they are two inter-dependent low-dimensional representations of the row and column objects derived from the original data matrix. Co-embedding is a unique and novel by-product of our algorithm, which accomplishes dimensionality reduction while preserving dependencies between rows and columns. None of partitional co-clustering algorithms is able to generate such an embedding, since they do not allow mixed membership to row and column clusters. To visualize the co-embedding, we apply ISOMAP [21] on $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ to further reduce the space to 2 dimensions.[3]

The results of co-embedding for users and movies on binarized Movielens dataset are shown in Figure 9(a) and 9(c). Each point in the figure denotes one user/movie. We mark three clusters with red, blue and green for users and movies respectively; other points are colored pink. By visualization, we can see how the users/movies are scattered in the space, where the clusters are located, and how far one cluster is from another, etc. Such information goes far beyond clusters of objects only. In addition, we choose several points from the co-embedding to look at their properties. In Figure 9(a) and 9(c), we mark four users and four movies, and extract their "signatures". In general, we can use a variety of methods to generate signature. In our experiment, we do the following: For each user, among all the movies she rates "1", we get the number of movies in movie cluster 1-20 respectively. After normalization, this 20-dim unit vector is used as the signature for the user. Similarly, for each movie, among all the users giving it a rating of "1", we get the number

[3]An alternative approach would be to set k_1 and k_2 to 2, so that $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ are themselves 2 dimensional.

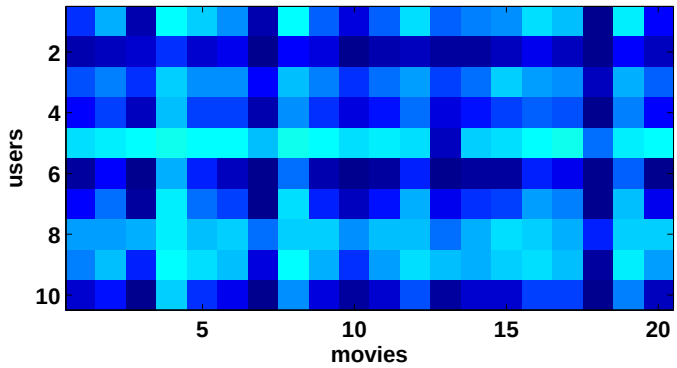


Figure 10: Co-cluster parameters for Movielens.

of users in user cluster 1-10 respectively. The normalized 10-dim unit vector is used as the signature for the movie. The signatures are shown in Figure 9(b) and 9(d) respectively. The numbers on the right are user/movie IDs corresponding to those marked points in co-embedding plots, showing where they are located. We can see that each signature is quite different from others in terms of the value on each component.

6 Conclusion

In this paper, we have proposed Bayesian co-clustering (BCC) which views co-clustering as a generative mixture modeling problem. BCC inherits the strengths and robustness of Bayesian modeling, is designed to work with sparse matrices, and can use any exponential family distribution as components of the generative model, thereby making it suitable for a wide range of matrices. Unlike existing partitional co-clustering algorithms, BCC generates mixed memberships for rows and columns, which seem more appropriate for a variety of applications. A key advantage of the proposed variational approximation approach for BCC is that it is expected to be significantly faster than a stochastic approximation based on sampling, making it suitable for large matrices in real life applications. Finally, the co-embedding obtained from BCC can be effectively used for visualization, subsequent predictive modeling, and decision making.

Acknowledgements: This work was supported by NASA under award NNX08AC36A.

References

[1] D. Agarwal and S. Merugu. Predictive discrete latent factor models for large scale dyadic data. In *KDD*, pages 26–35, 2007.

[2] E. Airoldi, D. Blei, S. Fienberg, and E. Xing. Mixed membership stochastic blockmodels. Technical report, arXiv:0705.4485v1 [stat.ME], 2007.

[3] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. Modha. A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. *JMLR*, 8:1919–1986, 2007.

[4] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *JMLR*, 6:1705–1749, 2005.

[5] A. Banerjee and H. Shan. Latent Dirichlet conditional naive-bayes models. In *ICDM*, pages 421–426, 2007.

[6] O. Barndorff-Nielsen. *Information and Exponential Families in Statistical Theory.* Wiley Publishers, 1978.

[7] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[8] Y. Cheng and G. Church. Biclustering of expression data. In *ISMB*, pages 93–103, 2000.

[9] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *JRSS*, 39(1):1–38, 1977.

[10] I. Dhillon, S. Mallela, and D. Modha. Information-theoretic co-clustering. In *KDD*, pages 89–98, 2003.

[11] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *PAMI*, 6(6):721–741, 1984.

[12] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *ICDM*, pages 625–628, 2005.

[13] J. Hartigan. Direct clustering of a data matrix. *JASA*, 67(337):123–129, 1972.

[14] C. Kemp, J. Tenenbaum, T. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, 06.

[15] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[16] Y. Kluger, R. Basri, J. Chang, and M. Gerstein. Spectral biclustering of microarray data: Coclustering genes and conditions. *Genome Research*, 13(4):703–716, 2003.

[17] G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions.* Wiley-Interscience, 1996.

[18] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368, 1998.

[19] K. Nowicki and T. Snijders. Estimation and prediction for stochastic blockstructures. *JASA*, 96(455):1077–1087, 2001.

[20] R. Redner and H. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195–239, 1984.

[21] J. Tenenbaum, V. Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

A Variational Inference

The lower bound of $\log p(X|\alpha_1, \alpha_2, \Theta)$ could be obtained from Eq. 3 and represented as

$$\begin{aligned} L = & E_q[\log p(\boldsymbol{\pi}_1|\alpha_1)] + E_q[\log p(\boldsymbol{\pi}_2|\alpha_2)] + E_q[\log p(\mathbf{z}_1|\boldsymbol{\pi}_1)] + E_q[\log p(\mathbf{z}_2|\boldsymbol{\pi}_2)] \\ & + E_q[p(X|\mathbf{z}_1, \mathbf{z}_2, \Theta)] \\ & - E_q[\log q(\boldsymbol{\pi}_1|\gamma_1)] - E_q[\log q(\boldsymbol{\pi}_2|\gamma_2)] - E_q[\log q(\mathbf{z}_1|\boldsymbol{\phi}_1)] - E_q[\log q(\mathbf{z}_2|\boldsymbol{\phi}_2)], \end{aligned}$$

We denote the lower bound with L for brevity. Each term in L could be expanded as in Table 3.

Term	Expression
$E_q[\log p(\boldsymbol{\pi}_1\|\alpha_1)]$	$\sum_{u=1}^{n_1}\sum_{i=1}^{k_1}(\alpha_{1i}-1)(\Psi(\gamma_{1ui})-\Psi(\sum_{l=1}^{k_1}\gamma_{1ul}))+n_1\log\Gamma(\sum_{i=1}^{k_1}\alpha_{1i})-n_1\sum_{i=1}^{k_1}\log\Gamma(\alpha_{1i})$
$E_q[\log p(\boldsymbol{\pi}_2\|\alpha_2)]$	$\sum_{v=1}^{n_2}\sum_{j=1}^{k_2}(\alpha_{2j}-1)(\Psi(\gamma_{2vj})-\Psi(\sum_{l=1}^{k_2}\gamma_{2vl}))+n_2\log\Gamma(\sum_{j=1}^{k_2}\alpha_{2j})-n_2\sum_{j=1}^{k_2}\log\Gamma(\alpha_{2j})$
$E_q[\log p(\mathbf{z}_1\|\boldsymbol{\pi}_1)]$	$\sum_{u=1}^{n_1}\sum_{i=1}^{k_1}m_u\phi_{1ui}(\Psi(\gamma_{1ui})-\Psi(\sum_{l=1}^{k_1}\gamma_{1ul}))$
$E_q[\log p(\mathbf{z}_2\|\boldsymbol{\pi}_2)]$	$\sum_{v=1}^{n_2}\sum_{j=1}^{k_2}m_v\phi_{2vj}(\Psi(\gamma_{2vj})-\Psi(\sum_{l=1}^{k_2}\gamma_{2vl}))$
$E_q[\log q(\boldsymbol{\pi}_1\|\boldsymbol{\gamma}_1)]$	$\sum_{u=1}^{n_1}\sum_{i=1}^{k_1}(\gamma_{1ui}-1)(\Psi(\gamma_{1ui})-\Psi(\sum_{l=1}^{k_1}\gamma_{1ul}))+\sum_{u=1}^{n_1}\log\Gamma(\sum_{i=1}^{k_1}\gamma_{1ui})-\sum_{u=1}^{n_1}\sum_{i=1}^{k_1}\log\Gamma(\gamma_{1ui})$
$E_q[\log q(\boldsymbol{\pi}_2\|\boldsymbol{\gamma}_2)]$	$\sum_{v=1}^{n_2}\sum_{j=1}^{k_2}(\gamma_{2vj}-1)(\Psi(\gamma_{2vj})-\Psi(\sum_{l=1}^{k_2}\gamma_{2vl}))+\sum_{v=1}^{n_2}\log\Gamma(\sum_{j=1}^{k_2}\gamma_{2vj})-\sum_{v=1}^{n_2}\sum_{j=1}^{k_2}\log\Gamma(\gamma_{2vj})$
$E_q[\log q(\mathbf{z}_1\|\phi_1)]$	$\sum_{u=1}^{n_1}\sum_{v=1}^{n_2}\sum_{i=1}^{k_1}\sum_{j=1}^{k_2}\delta_{uv}\phi_{1ui}\phi_{2vj}\log\phi_{1ui}$
$E_q[\log q(\mathbf{z}_2\|\phi_2)]$	$\sum_{u=1}^{n_1}\sum_{v=1}^{n_2}\sum_{i=1}^{k_1}\sum_{j=1}^{k_2}\delta_{uv}\phi_{1ui}\phi_{2vj}\log\phi_{2vj}$
$E_q[\log p(X\|\mathbf{z}_1, \mathbf{z}_2, \Theta)]$	$\sum_{u=1}^{n_1}\sum_{v=1}^{n_2}\sum_{i=1}^{k_1}\sum_{j=1}^{k_2}\delta_{uv}\phi_{1ui}\phi_{2vj}\log p_\psi(x_{uv}\|\theta_{ij})$

Table 3: Expression for terms in $L(\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \boldsymbol{\phi}_1, \boldsymbol{\phi}_2; \alpha_1, \alpha_2, \Theta)$

A.1 Inference

For inference, we first maximize L with respect to ϕ_{1u}. It is a constrained optimization with the constraint $\sum_{i=1}^{k_1}\phi_{1ui}=1$. We construct the Lagrangian by isolating the terms containing ϕ_{1ui} and adding the Lagrange multipliers:

$$\begin{aligned} L_{[\phi_{1u}]} = & \sum_{v=1}^{n_2}\sum_{i=1}^{k_1}\sum_{j=1}^{k_2}\delta_{uv}\phi_{1ui}\phi_{2vj}\Big(\Psi(\gamma_{1ui})+\Psi(\gamma_{2vj})-\Psi(\sum_{l=1}^{k_1}\gamma_{1ul})-\Psi(\sum_{l=1}^{k_2}\gamma_{2vj}) \\ & +\log p(x_{uv}|\theta_{ij})-\log\phi_{1ui}-\log\phi_{2vj}\Big)+\lambda_{1u}(\sum_{i=1}^{k_1}\phi_{1ui}-1)\ . \end{aligned}$$

By taking derivative w.r.t. ϕ_{1ui} and setting it to zero, we have:

$$\phi_{1ui}\propto\exp\left(\Psi(\gamma_{1ui})+\frac{\sum_{v=1}^{n_2}\sum_{j=1}^{k_2}\delta_{uv}\phi_{2vj}\log p(x_{uv}|\theta_{ij})}{m_u}\right)\ .$$

Similarly, for ϕ_{2v}, we have

$$\phi_{2vj}\propto\exp\left(\Psi(\gamma_{2vj})+\frac{\sum_{u=1}^{n_1}\sum_{i=1}^{k_1}\delta_{uv}\phi_{1ui}\log p(x_{uv}|\theta_{ij})}{m_v}\right)\ .$$

Next, we maximize L with respect to γ_{1u}. The terms containing γ_{1u} are:

$$L_{[\gamma_{1u}]}=\sum_{i=1}^{k_1}\big(\Psi(\gamma_{1ui})-\Psi(\sum_{l=1}^{k_1}\gamma_{1ul})\big)(\alpha_{1i}+m_u\phi_{1ui}-\gamma_{1ui})+\sum_{i=1}^{k_1}\log\Gamma(\gamma_{1ui})-\log\Gamma(\sum_{i=1}^{k_1}\gamma_{1ui})\ .$$

Again, taking derivative w.r.t. γ_{1ui} and setting it to zero yields:

$$\gamma_{1ui} = \alpha_{1i} + m_u \phi_{1ui} \ .$$

Similarly,

$$\gamma_{2vj} = \alpha_{2j} + m_v \phi_{2vj} \ .$$

A.2 Parameter Estimation

For parameter estimation, we first maximize L with respect to Θ, which is specific to the generative model we choose. In Gaussian case, we have:

$$L_{[\mu,\sigma]} = \sum_{u=1}^{n_1}\sum_{v=1}^{n_2}\sum_{i=1}^{k_1}\sum_{j=1}^{k_2} \delta_{uv}\phi_{1ui}\phi_{2vj}\left(-\frac{(x_{uv}-\mu_{ij})^2}{2\sigma_{ij}^2} - \log\sqrt{2\pi\sigma_{ij}^2}\right)$$

We take derivative respect to μ_{ij} and σ_{ij}^2, and set it to zero to get:

$$\begin{aligned}
\mu_{ij} &= \frac{\sum_{u=1}^{n_1}\sum_{v=1}^{n_2}\delta_{uv}x_{uv}\phi_{1ui}\phi_{2vj}}{\sum_{u=1}^{n_1}\sum_{v=1}^{n_2}\delta_{uv}\phi_{1ui}\phi_{2vj}} \\
\sigma_{ij}^2 &= \frac{\sum_{u=1}^{n_1}\sum_{v=1}^{n_2}\delta_{uv}(x_{uv}-\mu_{ij})^2\phi_{1ui}\phi_{2vj}}{\sum_{u=1}^{n_1}\sum_{v=1}^{n_2}\delta_{uv}\phi_{1ui}\phi_{2vj}} \ .
\end{aligned}$$

To estimate α_1 and α_2, the terms containing α_1 are:

$$L_{[\alpha_1]} = \sum_{u=1}^{n_1}\sum_{i=1}^{k_1}(\alpha_{1i}-1)(\Psi(\gamma_{1ui}) - \Psi(\sum_{l=1}^{k_1}\gamma_{1ul})) + n_1\log\Gamma(\sum_{i=1}^{k_1}\alpha_{1i}) - n_1\sum_{i=1}^{k_1}\log\Gamma(\alpha_{1i}).$$

Taking derivation w.r.t. α_{1i} yields:

$$\frac{\partial L}{\partial \alpha_{1i}} = \sum_{u=1}^{n_1}\left(\Psi(\gamma_{1ui}) - \Psi(\sum_{l=1}^{k_1}\gamma_{1ul})\right) + n_1\left(\Psi(\sum_{i=1}^{k_1}\alpha_{1i}) - \Psi(\alpha_{1i})\right) \ .$$

The derivation depends on $\{\alpha_{1l}, [l]_1^{k_1}, l \neq i\}$, so we could not obtain a close form solution for α_{1i}. Following [7], we adopt Newton-Raphson algorithm to find α_{1i} iteratively, where

$$\frac{\partial L}{\partial \alpha_{1i}\alpha_{1i}} = n_1\Psi'(\sum_{i=1}^{k_1}\alpha_{1i}) - n_1\Psi'(\alpha_{1i}) \tag{11}$$

$$\frac{\partial L}{\partial \alpha_{1i}\alpha_{1l}} = n_1\Psi'(\sum_{i=1}^{k_1}\alpha_{1i}) \qquad (l \neq i) \ . \tag{12}$$

So the Hessian matrix has (11) on diagonal and (12) everywhere else. Given gradient $g(\cdot)$ and Hessian $H(\cdot)$, Newton-Raphson finds the optimal solution by using the following updating function:

$$\alpha_1' = \alpha_1 + H(\alpha_1)^{-1}g(\alpha_1) \ .$$

However, what we are actually dealing with is a constrained optimization problem since $\alpha_1 > 0$. Iterating using Newton-Raphson without constraint sometimes takes the updated value outside the

feasible range. Therefore, we are using an adaptive line search in the direction of of updating by iterating:

$$\alpha_1' = \alpha_1 + \eta H(\alpha_1)^{-1} g(\alpha_1) \ .$$

Multiplying the second term by η, we are performing a line search to prevent α_1 to go out the feasible range ($\alpha_1 > 0$). At each updating step, we first assign η to be 1. If α_1' is inside the feasible range, we go on to the next step, otherwise, we decrease η by a factor of 0.5 until α_1' becomes valid. The objective function is guaranteed to be improved since we are not change the update direction but only the scale. Similar strategy is performed on α_2.