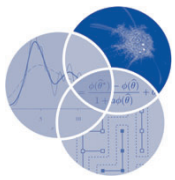
Advanced Review



Bayesian co-clustering

Carlotta Domeniconi[1]* and Kathryn Laskey[2]

Co-clustering means simultaneously identifying natural clusters in different kinds of objects. Examples include simultaneously clustering customers and products for a recommender application; simultaneously clustering proteins and molecules in microbiology; or simultaneously clustering documents and words in a text mining application. Important insights into a problem can be gained by understanding the interactions between clusters for the different kinds of objects. This paper considers Bayesian models for co-clustering. The Bayesian approach begins by developing a model for the data generating process, and inverting that model through Bayesian inference to infer cluster membership, learn characteristics of the clusters, and fill in missing observations. We consider a basic Bayesian clustering model and several extensions to the model. Experimental evaluations and comparisons among the clustering methods are presented.

How to cite this article:
WIREs Comput Stat 2015, 7:347–356. doi: 10.1002/wics.1359

Keywords: co-clustering; ensemble methods; Bayesian data mining; Bayesian nonparametrics; Markov chain Monte Carlo

INTRODUCTION

A fundamental problem in data analysis is to group objects into natural clusters. A cluster is a group of objects that are similar to each other and dissimilar to objects outside the cluster (c.f., Ref 1). Often, clustering is performed as a preliminary exploratory step before further modeling and analysis. In many applications, it is useful to cluster different kinds of objects simultaneously. For example, a recommender system might simultaneously cluster customers and products, so that customers in the same customer cluster have similar patterns for purchasing products in the same product cluster. A microbiologist might simultaneously cluster genes and experimental conditions so that genes in the same gene cluster have similar expression levels under conditions in the same condition cluster. A document analyst might simultaneously cluster documents and words so that documents in the same document cluster tend to contain words in the same word cluster. Understanding the interactions between clusters for the different kinds of objects can give important insights into the problem under study. For example, simultaneously grouping molecules according to their binding patterns with proteins and proteins according to how they interact with molecules can yield insight into the underlying biological processes, ultimately yielding better drug designs.

In each of the above examples, we can arrange the data in a matrix, where rows represent one kind of object (e.g., customers, genes, documents), columns represent another kind of object (e.g., products, experimental conditions, words), and each entry represents data collected for the row and column object pair (e.g., whether the customer purchased the product; expression pattern of the gene under the experimental conditions; whether the word appears in the document). Co-clustering can be thought of as reordering the rows and columns of the data matrix to form homogeneous blocks; the co-clusters are the rows/columns whose intersections form the homogeneous blocks. To illustrate, the left-hand side of Figure 1 shows a data matrix, for which a darker color is used to indicate larger numbers. No pattern is immediately apparent, but if the rows and columns are rearranged as

*Correspondence to: carlotta@cs.gmu.edu

[1]Department of Computer Science, George Mason University, Fairfax, VA, USA

[2]Systems Engineering and Operations Research (SEOR), George Mason University, Fairfax, VA, USA

Conflict of interest: The authors have declared no conflicts of interest for this article.

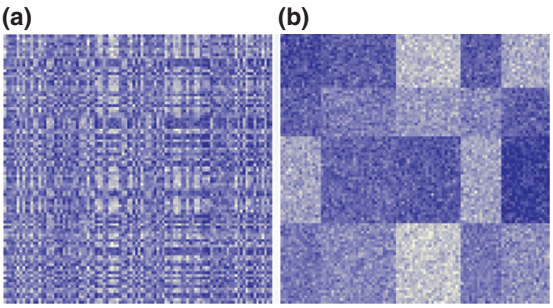


FIGURE 1 | Original data matrix (a); reordered to reveal co-clusters (b).

shown on the right, row and column clusters become apparent.

BACKGROUND

Co-clustering, also known as biclustering, was first studied by Hartigan[2] and has been investigated in a variety of application areas (e.g., Refs 3–5). Many algorithms for co-clustering have appeared in the literature, e.g., Refs 3, 6

This paper studies Bayesian methods of co-clustering. The Bayesian approach begins with a formal model of the data-generating process, and uses the observations to update this model. In co-clustering, the observations consist of a two-dimensional matrix x whose rows and columns refer to objects and whose entries represent data collected for the corresponding row and column objects. For example, the entry x_{rc} might contain the rth person's rating of the cth product, or an indicator of whether the cth word appears in the rth document. The aim of co-clustering is to find natural clusters such that entries of x for objects in the same row and column cluster tend to be similar. The observations $\{x_{rc}\}$ are used to update prior knowledge about membership of objects in row and column clusters and parameters governing the cluster characteristics.

A Bayesian approach to co-clustering has several advantages. The model considered below uses soft partitions, whereby each object can belong in some degree to multiple clusters. This is more realistic than traditional hard partitioning methods in which each object is assigned to exactly one cluster. Bayesian co-clustering is based on a clearly interpretable generative model that defines a full joint probability distribution on unknown parameters, latent row and column clusters, and observations. On the basis of this joint distribution, Bayesian inference provides a theoretically principled foundation for algorithms to learn the parameters of the generative model and infer the unobserved cluster memberships. Because it is based on a full joint distribution, the Bayesian approach naturally handles missing observations, a necessary requirement for many applications. For example, each customer typically rates only a small subset of products, yielding a substantial fraction of missing ratings. A recommender system unable to handle missing ratings would be useless. Bayesian co-clustering can discover customer and product co-clusters using only a sample of product ratings from each customer, and then use the co-clusters to predict a customer's ratings for products he/she has not rated. Further, it is well known that Bayesian inference protects naturally against overfitting, a problem that plagues many machine learning algorithms. Protecting against overfitting is especially important for models that have a large number of parameters and/or unobserved variables, as do the models considered in this paper.

A BASIC BAYESIAN CO-CLUSTERING MODEL

In this section, we consider a basic co-clustering model, which is later extended in several directions. Prior information about the data-generating process is incorporated into the model by defining a sampling process for generating entries of the data matrix x. Such a generative model is typically represented as a graphical probability model, a standard and convenient way to visualize a complex joint probability distribution over many variables. Figure 2 depicts a graphical probability model for the generative process taken from Ref 7; closely related models are given in Refs 8, 9. Each of the circles in the graphical model represents a quantity that is sampled. The circle for x is dark, indicating that the entries of the data matrix are observed. The other circles are light, indicating that they represent unobserved quantities. The arrows depict dependencies: a quantity at the head of an arrow is sampled in a way that depends on the result of sampling the quantity at the tail of the arrow. The rectangles, called plates, denote repeated samples, with the number in the lower right-hand corner indicating the number of repetitions.

The variables in the model consist of the observations x, the latent cluster memberships z, the parameters π, θ of the distributions from which z and x are sampled, and the hyperparameters α, β, or parameters of the distributions from which the parameters π and θ are sampled. We can walk through the generative process by following the arrows in the graphical model. Each quantity is sampled after all its predecessors in the graph have been sampled, using a distribution that

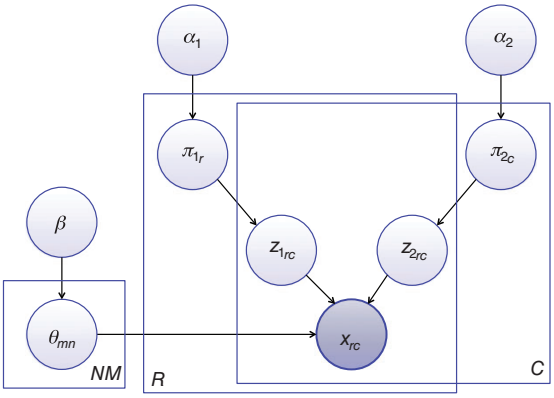


FIGURE 2 | Graphical model for Bayesian co-clustering.

depends on its immediate predecessors. Sampling proceeds as follows:

1. Select hyperparameters: α_1, which governs sampling of the latent row cluster memberships; α_2, which governs sampling of the latent column cluster memberships; and β, which governs sampling of the co-cluster characteristics.
2. Select co-cluster parameters $\{\theta_{mn}{:}m=1, \ldots, M, n=1, \ldots, N\}$ for the M row clusters and N column clusters, from a parametric family of probability distributions with parameter β. The co-cluster parameter θ_{mn} governs sampling of observations in the (mn)th block of the matrix. In a gene expression problem, θ_{mn} might denote the mean and standard deviation of the gene expression level for the block. In a text mining problem, θ_{mn} might denote the probability that a given word in the row cluster appears in a given document in the column cluster. For a given application, the distribution for θ is chosen from a parametric family appropriate for the type of data in the matrix x.
3. Choose row cluster membership probability vectors $\{\pi_{1_r} : r = 1, \ldots, R\}$ for the R rows of the matrix, from a probability distribution that depends on the parameter α_1. Each probability vector π_{1_r} consists of M non-negative numbers summing to 1. It governs sampling of the row cluster latent variables, giving higher probability to some row clusters than others. Thus, π_{1_r} can be thought of as defining the strength of membership of the rth object in each of the M row clusters. The probability vectors π_{1_r} are drawn from a Dirichlet distribution with parameter α_1, a parametric family with convenient mathematical properties for sampling probability distributions.
4. Choose column cluster membership probability vectors $\{\pi_{2_c} : c = 1, \ldots, C\}$ for the C columns of the matrix, from a probability distribution that depends on the parameter α_2. Each probability vector π_{2_c} consists of N non-negative numbers summing to 1. It governs sampling of the column cluster latent variables, giving higher probability to some column clusters than others. Thus, π_{2_c} can be thought of as defining the strength of membership of the cth object in each of the M column clusters. The probability vectors π_{2_c} are drawn from a Dirichlet distribution with parameter α_2.
5. Choose the row and column cluster latent variables for each of the RC entries in the matrix. The latent row cluster $z_{1_{rc}}$ for the entry in row r and column c is drawn from the rth row cluster membership probability vector π_{1_r}. The latent column cluster $z_{2_{rc}}$ for the entry in row r and column c is drawn from the cth column cluster membership probability vector π_{2_c}.
6. Choose the entry x_{rc} from the observation distribution for its latent row and column cluster, that is, the distribution with parameter $\theta_{z_{1_{rc}} z_{2_{rc}}}$.

This sampling process generates a soft partition over row and column clusters. For each cell (r, c) of the data matrix, the entry x_{rc} is drawn from a distribution that depends on the row and column latent variables $z_{1_{rc}}$ and $z_{2_{rc}}$, which in turn are drawn according to their respective cluster membership probabilities π_{1_r} and π_{2_c}. These cluster membership probability vectors give greater preference to some clusters than others. That is, more of the latent row cluster variables $z_{1_{rc}}$ for the rth row of the matrix will, on average, have values to which π_{1_r} assigns higher probability. Thus, π_{1_r} represents the strength of membership of row object r in each of the row clusters. Similarly, π_{1_c} represents the strength of membership of column object c in each of the column clusters.

The model of Figure 2 defines a joint probability distribution

$$p(\alpha, \beta, \pi, \theta, z, x) = p(\alpha)\, p(\beta)\, p(\theta|\beta)\, p(\pi|\alpha)\, p(z|\pi)\, p(x|z) \quad (1)$$

over parameters, latent variables, and data. This distribution is a hierarchical Bayesian model in which hyperparameters α, and β are drawn independently of each other; θ is drawn from a distribution that depends on β; π is drawn from a distribution that depends on α;

z is drawn from a distribution that depends on π and θ; and x is drawn from a distribution that depends on z. The plate representation of Figure 2 is a convenient visualization of the hierarchical model.

The following section describes how this model is updated in the light of data to learn the row and column cluster membership probabilities π_1 and π_2 and the cluster characteristics θ, as well as to fill in missing entries in the data matrix.

INFERENCE

After the data matrix is observed, the distributions for hyperparameters, parameters and latent variables are updated using Bayesian inference. The posterior joint distribution of the hyperparameters (α, β), the parameters (π, θ), and the latent variables z, given the observations x, is obtained by applying Bayes rule:

$$p(\alpha, \beta, \pi, \theta, z|x) = \frac{p(\alpha)\,p(\beta)\,p(\theta|\beta)\,p(\pi|\alpha)\,p(z|\pi)\,p(x|z)}{p(x)} \quad (2)$$

Bayes rule is a principled means of combining evidence with prior information. For the model of Figure 2, the prior information includes the structure of the model and prior probability distributions that specify vague information about the parameters. After applying Bayes rule, the posterior distribution becomes concentrated on configurations of $(\alpha, \beta, \pi, \theta, z)$ that are consistent with the data matrix. Thus, the posterior distribution will place high probability on latent row and column cluster assignments $z_{1_{rc}}$ and $z_{2_{rc}}$ such that entries x_{rc} within the same row and column cluster are similar.

From the joint distribution, we can derive probabilities for many hypotheses of interest. For example, we can find the posterior probability π_{1_r} that row object r belongs to each of the row clusters. Similarly, we can find posterior membership probabilities π_{2_c} for column objects in column clusters. We can also find the a posteriori most likely value of θ_{mn}, the parameter governing characteristics of matrix entries representing interactions between row objects in the mth row cluster and column objects in the nth column cluster. Other posterior quantities of interest might include matrix blocks with the highest or lowest expected values, or differences in the expected value of entries in different blocks. All these quantities can be obtained from the joint probability distribution (2).

Another key problem is to predict missing entries of the data matrix x. Missing entries are common in many applications, as when each customer rates only some products, or binding patterns are observed for only some protein-molecule interactions. Equation (2) can be modified to find the posterior distribution of hyperparameters, parameters, latent variables, and unobserved data matrix entries conditional on the observed matrix entries. Clearly, the quality of the results will depend on the pattern of coverage, e.g., if a row has almost no entries, then inferences about its row cluster membership will be weak. Useful inferences can be drawn for parameters or latent variables for which there are sufficient data.

While the posterior distribution (2) is well-defined in theory, calculating posterior estimates of interest is very challenging computationally. Although exact inference is generally intractable, several approximate inference methods are available. The two most common classes of approximate inference methods are model simplification and stochastic approximation. The model simplification approach finds a simpler distribution $q(\alpha, \beta, \pi, \theta, z)$ that is as close as possible to the true posterior distribution $p(\alpha, \beta, \pi, \theta, z|x)$ but can be solved exactly. Stochastic approximation draws samples from the posterior distribution $p(\alpha, \beta, \pi, \theta, z|x)$ and uses the samples to estimate quantities of interest.

The most widely applied model simplification approach is called variational Bayesian inference (c.f., Ref 10). The key idea of variational inference is to approximate $p(\alpha, \beta, \pi, \theta, z|x)$ by a distribution $q(\alpha, \beta, \pi, \theta, z)$ chosen from a family of distributions with a simpler functional form. Generally, $q(\alpha, \beta, \pi, \theta, z)$ has a product form, where each factor in the product is a tractable distribution involving a subset of the unobserved variables, and all these subsets are disjoint. The variational algorithm then iterates over the factors, adjusting each one so it more closely matches the true posterior distribution given the data and the current estimate of the other factors. The algorithm cycles through the factors repeatedly until the improvement is less than a threshold. Under reasonable assumptions, this process is guaranteed to converge to the member $q^*(\alpha, \beta, \pi, \theta, z)$ of the tractable family that is closest to $p(\alpha, \beta, \pi, \theta, z|x)$.

Finding an appropriate family of tractable factored distributions can be a challenge. Once a suitable family has been identified, deriving the updating equations typically involves tedious mathematical manipulations. After the work of finding a family and deriving updating equations has been accomplished, however, execution of the resulting approximate inference algorithm is typically quite fast in comparison with sampling based approaches.

The most common class of stochastic sampling algorithms is called Markov chain Monte Carlo

(MCMC; c.f., Refs 11, 12). The most straightforward MCMC method is Gibbs sampling. Like variational inference, Gibbs sampling is an iterative process that repeatedly cycles through the unobserved variables. Unlike variational inference, the iterative step is stochastic. Gibbs sampling begins by assigning an arbitrary value to each unobserved variable. Then, at each step of the iterative process, the unobserved variable is replaced by a value sampled from its posterior distribution conditional on all the other (observed and unobserved) variables. Under reasonable assumptions, this process is guaranteed to converge to a sample from the joint posterior distribution, but convergence can be very slow. In fact, for the model of Figure 2, convergence is much too slow to be practical. However, a variant called collapsed Gibbs sampling brings MCMC into the range of feasibility. Collapsed Gibbs sampling works as follows. First, the hyperparameters are set to fixed values. Once this has been done, we can integrate over the parameters analytically, to obtain conditional distributions for each latent variable given the hyperparameters, the other latent variables, and the elements of the data matrix. Then Gibbs sampling can be performed over just the latent variables. If there are missing data, collapsed Gibbs sampling can be performed over both latent variables and missing observations. The values for the hyperparameters can be set as a tuning step, or an additional Gibbs sampling step over the hyperparameters can be added.

The standard variational Bayesian algorithm for the model of Figure 2 can give inaccurate results because it does not account for dependence between the latent variables z_1, z_2 and the parameters π_1, π_2, θ. Inspired by collapsed Gibbs sampling, Wang et al.[7] introduced a variational algorithm, called collapsed variational Bayes, which operates in the collapsed space where the parameters are integrated out of the log likelihood function.

EXPERIMENTS

Wang et al.[7] reported experiments using the model of Figure 2. Their experiments compared variational Bayesian inference, collapsed Gibbs sampling, and collapsed variational Bayesian inference. They considered two datasets. The first is MovieLens,[a] a movie recommendation dataset containing 100,000 ratings in a sparse data matrix for 1682 movies rated by 943 users. The second is Jester,[b] a joke rating dataset. The original Jester dataset contains 4.1 million ratings of 140 jokes from 73,421 users. Following,[8] a dense sub-matrix was selected of 1000 users who rated almost all jokes. The continuous ratings were transformed to binary form by coding all non-negative entries as 1 and all negative entries as 0. For both datasets, 25% of the observations were held out for testing.

TABLE 1 | Perplexity Comparison for Basic Co-Clustering Model

	Gibbs Sampling	Collapsed Variational Bayes	Variational Bayes
MovieLens	3.247	4.553	5.849
Binarized Jester	2.954	3.216	4.033

The model was trained using the three inference methods described in the previous Section. Results were compared using perplexity, a standard comparison metric for probabilistic methods. The perplexity of the probability model $p(x)$ on the data matrix x is defined as:

$$\text{Perp} = \exp\left(\frac{1}{\#_{rc}}\sum_{rc} -\log p\left(x_{rc}\right)\right), \qquad (3)$$

where the sum is over the observed entries of the data matrix and $\#_{rc}$ denotes the number of observed entries. The perplexity decreases as the likelihood of the observations increases. A lower perplexity on the training set means the model fits the training set well; a lower perplexity on the test set means the model does well at predicting unseen observations.

Table 1 shows perplexity results for the three inference methods on the two data sets. On both data sets, the worst performer is standard variational Bayesian inference, followed by collapsed variational Bayesian inference, followed by collapsed Gibbs sampling. However, both variational Bayesian methods converged after about 100 iterations, whereas collapsed Gibbs sampling required about 5000 iterations to reach convergence.

EXTENSIONS TO THE BASIC MODEL

We consider several extensions to the basic Bayesian co-clustering model. First, we consider the case in which the number of row and column clusters is not known a priori. Next, we consider an extension in which features of the row and column objects can be leveraged to help predict observations for new, as yet unobserved objects. Finally, we consider the case in which row and column clusters are not independent of each other.

Unknown Numbers of Clusters

The basic Bayesian co-clustering model requires that the number of row and column clusters be specified

in advance. Given that the aim of co-clustering is to discover latent clusters, it seems unreasonable to expect the number of clusters to be known. In practice, this issue is often addressed by treating the number of clusters as a tuning parameter and adjusting it to give a good fit to the data set. A more principled approach is to bring the number of clusters into the model, treating it as an unknown quantity to be discovered from the data. This can be achieved by replacing the Dirichlet prior distributions for π_1 and π_2 in Figure 2 with a distribution that places no bound on the number of clusters. Models with infinitely many parameters are called nonparametric models. A nonparametric Bayesian co-clustering (NBCC) model was considered by Meeds and Roweis.[13] Their model is shown on the left-hand side of Figure 3. In this model, the number of replications of the indices m and n of the co-cluster characteristics parameter θ_{mn} can grow without bound. In the figure, this is indicated by replacing the replication count MN in Figure 3 with the symbol ∞^2. Further, because the number of row and column clusters is unbounded, the parameters π_{1_r} and π_{2_c} are infinite-dimensional probability vectors. These vectors are drawn from a parametric family for sampling infinite-length non-negative real vectors whose entries sum to 1. For this purpose, Meeds and Roweis used the Pitman–Yor process, which generalizes the Dirichlet distribution used by the basic co-clustering model to sample finite-dimensional probability vectors.

The NBCC model of Figure 3 also differs from the basic co-clustering model in that each row or column object is assigned to a single row or column cluster, whereas the model of Figure 2 samples distinct row and column cluster latent variables for each entry of the matrix. The latter approach is a more flexible soft partitioning, but the former model is more easily extended to incorporate features, as described in the following section.

Incorporating Object Features

Thus far, we have considered models in which cluster discovery is based solely on the entries of the data matrix. Such methods are incapable of handling row or column objects that have not been seen before. For example, neither the basic model nor its nonparametric extension would be capable of predicting purchases for a customer with no prior purchases or a product that is new on the market. In many applications, additional information could be used to help predict data for new objects. In our recommendation example, we might have customer information such as age, gender, or occupation, as well as product information such as category, brand, or price. Such ancillary information can provide useful recommendations for new customers who have not yet made purchases, or for products that have not yet appeared on the market. A system that can incorporate this additional information can give useful generic initial recommendations that gradually become more tailored as more purchase data is obtained for the new customer and/or product.

The infinite hidden relational model (IHRM)[14] can be viewed as an extension of the NBCC model[13] to incorporate features. As shown on the right-hand side of Figure 3, IHRM adds feature variables y_{1_r} and y_{2_c}, as well as parameters ξ_{1_m} and ξ_{2_n} governing the feature distributions. The feature variables y_{1_r} and y_{2_c} represent features associated with row and column objects. According to this model, the row feature vector y_{1_r} for the rth row object is drawn from a parametric distribution with parameter ξ_{1_m}, where $m = z_{1_r}$ is the row cluster of the rth row object. Similarly, the column feature vector y_{2_c} for the cth column object is drawn from a distribution with parameter ξ_{2_n} that depends on the column cluster of the cth column object.

Wang et al.[15] performed simulation experiments to assess the impact of incorporating features into co-clustering. To emphasize the similarity to the NBCC model and to highlight the key difference, the models are called NBCC and feature-enriched NBCC, or FE-NBCC. The two models were compared using four data sets: the previously described MovieLens and Jester data sets, as well as two protein-molecule interaction data sets. For these experiments, the Jester joke ratings were uniformly discretized into 10 bins. The first protein interaction data set (MP1) contains data on 4051 interactions between 166 G-protein coupled receptor (GPCR) proteins and 2687 small molecules.[16] The second protein interaction data set (MP2) contains data on 7146 positive interactions between 154 proteins (this data set was not restricted to GPCR proteins) and 2876 molecules. Each of the data sets was divided into training and test data sets by first removing a randomly chosen subset of the row and column objects, to allow the algorithms to be evaluated on previously unseen objects, and then removing an additional randomly chosen set of entries in the matrix that remained. Each experiment was repeated five times. For inference, Wang et al. applied collapsed Gibbs sampling. Further details on how the experiments were conducted can be found in Ref 15.

Table 2 shows results from the experiment reported by Ref 15. The table shows average perplexity across the five runs, with standard deviations shown in parentheses. Not surprisingly, the algorithms performed similarly on previously seen objects, and

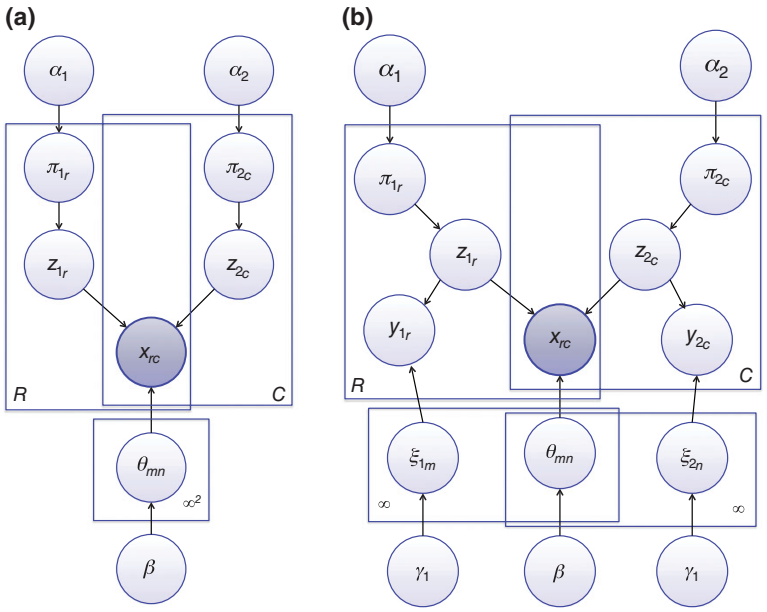


FIGURE 3 | Generative models for nonparametric Bayesian co-clustering (NBCC) (a) and feature-enriched nonparametric Bayesian co-clustering (FE-NBCC) (b).

TABLE 2 | Perplexity Comparison for Basic Co-Clustering Model

		NBCC	FE-NBCC
MovieLens	Row and column observed	3.327 (0.020)	3.344 (0.021)
	Row or column unseen	4.427 (0.047)	3.892 (0.026)
Jester (10 bins)	Row and column observed	17.111 (0.031)	17.125 (0.040)
	Row and column unseen	19.322 (0.025)	17.836 (0.053)
MP1	Row and column observed	1.430 (0.011)	1.435 (0.024)
	Row and column unseen	8.845 (0.011)	1.453 (0.026)
MP2	Row and column observed	1.484 (0.013)	1.489 (0.023)
	Row and column unseen	7.987 (0.011)	1.509 (0.024)

incorporating feature data resulted in substantially better performance for previously unseen objects.

Representing Non-Grid Co-Cluster Patterns

Most co-clustering algorithms assume that co-clusters are formed as the product of row and column clusters. This assumption of variation independence is not appropriate in many domains. For example, there might be a cluster of movies that elicits similar rating patterns from a wide variety of viewers, while another group of movies might have several clusters of users with distinct rating patterns. That is, how to cluster movies might depend on the user, and how to cluster users might depend on the movie. Thus, the rectangular block pattern exhibited by the basic co-clustering model might not capture all the useful structure in the data.

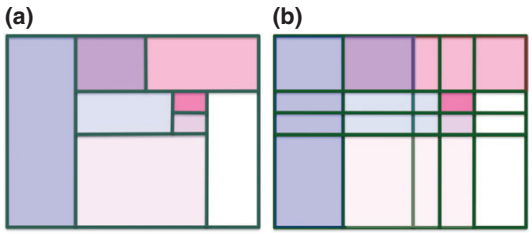


FIGURE 4 | Hierarchical axis-aligned partition (a) and corresponding grid partition (b).

We can incorporate more structure into the learned co-clusters if our prior distribution over co-clusters relaxes the requirement that co-clusters form a rectangular grid pattern. The Mondrian process,[17] named after the Dutch abstract painter Piet Mondrian, samples the kind of axis-aligned matrix partitions shown in Figure 4. As shown to the right

TABLE 3 | Perplexity Comparison for Co-Clustering Ensembles

		Base Co-Clustering	Co-Clustering Ensembles
MovieLens	Grid Partition	2.908 (0.055)	2.707 (0.060)
	Axis-Aligned Partition	2.553 (0.057)	2.436 (0.072)
Discretized Jester	Grid Partition	20.174 (0.219)	18.092 (0.458)
	Axis-Aligned Partition	16.144 (0.227)	14.036 (0.409)

of the figure, we can create a rectangular grid to represent these clusters, but this creates an unnecessary proliferation of clusters and fails to exploit key aspects of the problem structure.

The Mondrian process is a nonparametric model, in that the number of co-clusters is unbounded. The generative process begins with a rectangular matrix. A random choice is made whether to terminate sampling without further sub-division, to make a horizontal cut between two randomly chosen rows, or to make a vertical cut between two randomly chosen columns. If a cut is made, the process then repeats for each of the newly formed sub-rectangles.

The full generative model for Mondrian process co-clustering first randomly reorders the row and column objects, applies the Mondrian process to generate an axis-aligned partition of the reordered data matrix, chooses a parameter for the data distribution in each block of the matrix, and finally generates data according to the parameterized data distribution.

Inference for a Mondrian process consists of reconstructing the row and column ordering that reveals the axis-aligned partition, discovering the axis-aligned partition, and then learning the parameters of the data distribution for each block of the partition. Gibbs sampling cannot be used for Mondrian process inference because the required conditional distributions cannot be obtained in closed form. Instead, a generalization of Gibbs sampling called reversible jump Metropolis–Hastings (MH) sampling is used. Like Gibbs sampling, MH sampling proceeds through the variables one by one, sampling each in a way that depends on the values of the other variables. But whereas Gibbs sampling uses the conditional distribution of the sampled variable given the other variables, MH uses any distribution from which samples can be easily drawn. Then the sampled value is randomly either accepted or rejected according to a rule constructed to ensure convergence to the correct posterior distribution.

CO-CLUSTERING ENSEMBLES

Ensemble methods have become increasingly popular in machine learning and data mining as a way of improving robustness and increasing predictive performance. Ensemble clustering methods take as input a set of base clusterings and produce a consensus clustering. Recent work has shown that ensemble methods can identify robust consensus clusterings (e.g., Refs 18–20). Because they require only the clustering results and not the raw data, clustering ensembles provide a convenient approach to knowledge reuse.

There are two components involved in clustering ensemble techniques: how to generate diverse partitions (base clusterings), and how to combine the input base clusterings into a consensus clustering. Diverse partitions are typically generated by using different clustering algorithms,[21] or by applying a single algorithm with different parameter settings, possibly in combination with data or feature sampling. One way in which multiple clusterings can arise is through different local optima of a single base clustering algorithm. Rather than selecting a single local optimum, the ensemble approach recognizes that each local optimum may contribute its own distinct perspective, and that all local optima should contribute to the formation of the consensus clustering.

For the co-clustering problem, Wang et al.[22] considered two nonparametric Bayesian models, one based on the NPCC model and the other based on the Mondrian process. For each model, an ensemble of co-clusterings was generated by starting the inference algorithm at multiple initial points, thus generating a set of local optima of the base co-clustering algorithm. These base co-clusterings were used as input to the co-clustering ensemble algorithm. The algorithms were evaluated on the MovieLens and discretized Jester data sets. 25% of the observations were held out for testing. Results are shown in Table 3. The left-hand column shows the average perplexity for each of the base co-clusterings; the right-hand column shows perplexity for the ensemble method. Each result is averaged over multiple trials, with standard deviations shown in parentheses. Clustering ensembles perform better than the average of the base clusterings, and the Mondrian process model with its axis-aligned partitioning performs better than the standard axis-aligned partitioning. These differences are greater than can be accounted for by sampling error.

CONCLUSION

Co-clustering, or simultaneously finding natural clusters in different kinds of objects, has important applications in diverse fields such as text mining, computational biology, and recommender systems. This paper focuses on a Bayesian perspective to the problem of co-clustering and discusses a variety of models that capture important characteristics of real-world scenarios. A basic Bayesian model was first presented and applied to several data sets. The model was then extended to enable the number of clusters to be discovered from data. A further extension used features to allow co-clustering and prediction of observations for previously unseen objects. The introduction of the Mondrian process enabled modeling the inter-dependencies between row and column clusters. Finally, ensemble methods were integrated for learning a consensus co-clustering from a set of base co-clusterings, thereby achieving better local optima. Empirical results were presented for the basic model and its extensions.

NOTES

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

ACKNOWLEDGMENTS

This work was in part supported by NSF CAREER Award IIS-0447814.

REFERENCES

1. Everitt BS, Landau S, Leese M, Stahl D. *Cluster Analysis*. 5th ed. Chichester, UK: Wiley; 2011.
2. Hartigan JA. Direct clustering of a data matrix. *J Am Stat Assoc* 1972, 67:123–129.
3. Madeira SC, Oliveira AL. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Trans Comput Biol Bioinform* 2004, 1:24–45.
4. Dhillon IS. Co-clustering documents and words using bipartite spectral graph partitioning. In: *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '01*. New York: ACM; 2001, 269–274.
5. Bin X, Jiajun B, Chen C, Cai D. An exploration of improving collaborative recommender systems via user-item subgroups. In: *Proceedings of the 21st International Conference on World Wide Web, WWW '12*. New York, NY: ACM; 2012, 21–30.
6. Dhillon IS, Mallela S, Modha DS. Information-theoretic co-clustering. In: *KDD '03: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: ACM; 2003, 89–98.
7. Wang P, Domeniconi C, Laskey K. Latent Dirichlet Bayesian co-clustering. In *Proceedings of the European Conference on Machine Learning and Principles and Practise of Knowledge Discovery in Databases*, Bled, Slovenia, September 2009.
8. Shan H, Banerjee A. Bayesian co-clustering. In: *IEEE International Conference on Data Mining (ICDM)*. Washington DC: IEEE Press; 2008.
9. Shafiei MM, Milios EE. Latent Dirichlet co-clustering. In: *Sixth International Conference on Data Mining, 2006. ICDM '06*. Washington DC: IEEE Press; 2006, 542–551.
10. Fox CW, Roberts SJ. A tutorial on variational Bayesian inference. *Artif Intell Rev* 2012, 38:85–95.
11. Green PJ. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika* 1995, 82:711–732.
12. Andrieu C, de Freitas N, Doucet A, Jordan MI. An introduction to MCMC for machine learning. *Mach Learn* 2003, 50:5–43.
13. Meeds E, Roweis S. Nonparametric Bayesian biclustering. Technical Report UTML TR 2007–001, Department of Computer Science, University of Toronto, 2007.
14. Xu Z, Tresp V, Yu K, Kriegel H-P. Infinite hidden relational models. In *Proceedings of the 22nd International Conference on Uncertainty in Artificial Intelligence (UAI)*, MIT, Cambridge, MA, 2006.
15. Wang P, Domeniconi C, Rangwala H, Laskey KB. Feature enriched nonparametric Bayesian co-clustering. In: Tan P-N, Chawla S, Ho CK, Bailey J, eds. *Advances in Knowledge Discovery and Data Mining*. Lecture Notes in Computer Science, vol. 7301. Berlin/Heidelberg: Springer; 2012, 517–529.
16. Jacob L, Hoffmann B, Stoven V, Vert J-P. Virtual screening of GPCRs: an in silico chemogenomics approach. *BMC Bioinform* 2008, 9:363.

17. Roy DM, Teh YW. The Mondrian process. In: *Advances in Neural Information Processing Systems (NIPS)*, vol. 21. Cambridge, MA: MIT Press; 2008.
18. Topchy A, Jain AK, Punch W. Clustering ensembles: models of consensus and weak partitions. *IEEE Trans Pattern Anal Machine Intell* 2005, 27:1866–1881.
19. Strehl A, Ghosh J. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *J Mach Learn Res* 2003, 3:583–617.
20. Kuncheva LI. Experimental comparison of cluster ensemble methods. In: *International Conference on Information Fusion*. Washington DC: IEEE Computer Society; 2006, 1–7.
21. Greene D, Tsymbal A, Bolshakova N, Cunningham P. Ensemble clustering in medical diagnostics. In: *IEEE Symposium on Computer-Based Medical Systems*. Washington DC: IEEE Computer Society; 2004, 576–581.
22. Wang P, Laskey KB, Domeniconi C, Jordan M. Nonparametric Bayesian co-clustering ensembles. In: *SIAM International Conference on Data Mining*, 2011, 331–342.

