BAYESIAN STATISTICS

A non-parametric approach to modeling overlapping clusters

Jean-Baptiste REMY, Robin BEAUDET, Samuel RITCHIE

30 janvier 2018

Professor : **Arnak Dalalyan**

Table des matières

1 Introduction **3**

2 The mixture model **3**

3 Overlapping Mixture Models **4**

4 Infinite Overlapping Mixture Models via the IBP **6**

4.1 The idea behind the Indian Buffet Process 6

4.2 Computing the Z matrix 7

5 IOMM Learning **8**

6 Performances on toy datasets **9**

6.1 Binary data 10

6.2 Gaussian clusters 10

7 Applying overlapping clustering methods to gene expression data **11**

7.1 Motivations 11

7.2 An alternative approach : the Plaid biclustering algorithm 12

7.3 Dataset used 13

7.4 Results 13

7.4.1 Plaid algorithm 13

7.4.2 IOMM 14

8 Conclusion **14**

9 References **16**

10 Appendices **17**

10.1 Proof of Beta conjugate prior to Bernoulli likelihood 17

10.2 Equivalence of binary matrices 17

10.3 Results on the toy datasets 18

10.4 Plaid iterative algorithm 19

1 Introduction

Grouping data in such a way that objects in the same cluster are more similar to each other than to those in other clusters is a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis or bioinformatics.

Usually, clustering analysis rely on the hypothesis that each object belongs to one cluster. There are many applications for which this is not true. For instance, if we wish to cluster music, it would be true to state that Pink Floyd belongs to the clusters "Rock", "Progressive rock" and "Psychedelic rock". Another example in biostatistics would be to cluster gene families. Again, a particular gene may belong to more than one family as it may have several functions.

This modeling is also known as *overlapping clusters* in the sense that clusters may not need to be mutually exclusive as it is the case in many common approaches. This paper[1] aims to develop a model for overlapping clusters, starting from the well-known mixture model which can be found in the book *Pattern Recognition and Machine Learning*[2] (Springer, 2006) written by Bishop.

2 The mixture model

Consider K mutually exclusive clusters, where cluster j can be represented by a set of parameters θ_j, and consider also a set of individuals (x_i). For each cluster j, we can define a mixing parameter π_j, which is the weighting for distribution of cluster j. Thus, we have $\sum_j \pi_j = 1$ as the cluster are mutually exclusive.

Denoting the $p_j(x_i|\theta_j)$ the density for cluster j at point x_i, the conditional distribution is

$$p(x_i|\Theta) = \sum_{j=1}^{K} \pi_j p_j(x_i|\theta_j) \tag{1}$$

where Θ is the set of all the clusters parameters.

Let us introduce a K-dimensional binary random variable z in which a particular element z_k is equal to 1 and all other elements are equal to 0. Thus, the values of z_k satisfy $z_k \in \{0,1\}$ and $\sum_k z_k = 1$. Therefore, there are K possible states for the vector z according to which element is 1.

Using this variable z, we have :

$$p(x_i|\Theta) = \sum_{z_i} p(x_i, z_i|\Theta) = \sum_{z_i} p(z_i)p(x_i|z_i,\Theta) \tag{2}$$

and :

$$p(x_i|z_i,\Theta) = \prod_{j=1}^{K} p_j(x_i|\theta_j)^{z_{ij}} \tag{3}$$

Hence, the mixture model becomes :

$$p(x_i|\Theta) = \sum_{z_i} p(z_i) \prod_{j=1}^{K} p_j(x_i|\theta_j)^{z_{ij}} \tag{4}$$

where $\mathbf{P}(z_{i1} = 0, ..., z_{ij} = 1, z_{iK} = 0) = \pi_j$, which is the probability that point i belongs to cluster j. As the model assumes mutually exclusive clusters, it is straightforward to see that $\sum_j z_{ij} = 1$.

1. Katherine A. Heller, Zoubin Ghahramani. A Nonparametric Bayesian Approach to Modeling Overlapping Clusters. Journal of Machine Learning Research 2 :187-194, January 2007.
2. C. Bishop. Pattern Recognition and Machine Learning, Chapter 9. Springer, 2006.

From mutually exclusive to overlapping clusters

We can modify this simple framework to create a model that introduces overlapping clusters.

First, we need to remove the condition that is linked to the mutually exclusive clusters in order to allow overlaps. That is, we need to remove the restriction $\sum_j z_{ij} = 1$, which will lead to 2^K possible clusters[3].

Then, we would like to take into account the fact that we do not have any *a priori* about the number of clusters. This can be done by extending the number of clusters K to infinity using the Indian Buffet Process (IBP)[4]. The underlying idea is that the data itself will determine how many clusters are needed.

We now describe with more details the model.

3 Overlapping Mixture Models

As we said previously, we remove the constraint $\sum_j z_{ij} = 1$. To do this, we define the binary vector $z_i = (z_{i1}, ..., z_{iK})$ where $z_{ik} = 1$ if point i belongs to cluster k. Thus, if a point belongs to several clusters, each corresponding component in vector z_i will be nonzero[5].

We model the distributions of overlapping clusters by multiplying the distributions of the concerned clusters, up to a normalizing constant. As a consequence, for exponential families, if we represent two clusters by their density probability distributions, the overlapping cluster is represented by the region where the two distributions overlap (see next paragraph).

Denoting θ_k the set of parameters of cluster k, the conditional distribution of a point x_i is now given by :

$$p(x_i|z_i, \Theta) = \frac{1}{c} \prod_k p_k(x_i|\theta_k)^{z_{ik}} \tag{5}$$

where $c = \sum_{x \in \mathcal{X}} \left(\prod_k p_k(x_i|\theta_k)^{z_{ik}} \right)$ is the normalizing constant.

Using exponential families

Calculus are quite simple if we model clusters to be part of exponential families. Indeed, if

$$p_k(x_i|\theta_k) = g(x_i) f(\theta_k) e^{s(x_i)^T \phi(\theta_k)} \qquad \forall k \tag{6}$$

then :

$$\begin{aligned} p(x_i|z_i, \Theta) &= \frac{1}{c} \prod_k p_k(x_i|\theta_k)^{z_{ik}} & (7) \\ &= \frac{1}{c} \prod_k \left(g(x_i) f(\theta_k) e^{s(x_i)^T \phi(\theta_k)} \right)^{z_{ik}} & (8) \\ &= \frac{g(x_i)^{\sum_k z_{ik}}}{c} \Big[\prod_k f(\theta_k) \Big] e^{s(x_i)^T \left(\sum_k z_{ik} \phi(\theta_k) \right)} & (9) \end{aligned}$$

Therefore, $p(x_i|z_i\Theta)$ is of the form $p(x_i|z_i\Theta) = G(x_i) F(\theta_k) e^{S(x_i)^T \Phi(\theta_k)}$, which corresponds to the distribution of an exponential family.

3. These new clusters integrate both the original clusters and the overlapping clusters
4. Griffith and Ghahramani, 2006
5. This is different from the previous vector z_i which allowed only one component to be nonzero.

The special case of Gaussian clusters

Gaussian distributions are part of exponential families. Indeed, if Gaussian cluster k has mean μ_k and covariance Σ_k, then we have :

$$\begin{aligned} p(x_i|z_i,\mu,\Sigma) &= \frac{1}{c}\exp\left[-\frac{1}{2}\Big(x_i^T\Big(\sum_k z_{ik}\Sigma_k^{-1}\Big)x_i - 2x_i^T\Big(\sum_k z_{ik}\Sigma_k^{-1}\mu_k\Big) + \sum_k z_{ik}\mu_k^T\Sigma_k^{-1}\mu_k\Big)\right] \qquad (10) \\ &= \frac{1}{c}\exp\left[-\frac{1}{2}\Big(x_i^T S^{-1}x_i - 2x_i^T m + \sum_k z_{ik}\mu_k^T\Sigma_k^{-1}\mu_k\Big)\right] \qquad (11) \end{aligned}$$

with $S^{-1} = \sum_k z_{ik}\Sigma_k^{-1}$ and $m = \sum_k z_{ik}\Sigma_k^{-1}\mu_k$

Therefore, setting $\tilde{\Sigma} = S$ and $\tilde{\mu} = Sm$, we have :

$$x_i|z_i,\mu,\Sigma \sim \mathcal{N}\Big(\tilde{\mu},\tilde{\Sigma}\Big) \qquad (12)$$

The special case of multivariate Bernoulli clusters

For binary data $x_i \in \mathbf{R}^d$ (i.e. $x_{id} \in \{0,1\}$) and multivariate Bernouilli clusters ($\mathbf{P}_k(x_{id} = 1|\theta_k) = \theta_{kd}$), we get :

$$p(x_i|z_i,\Theta) = \frac{1}{c}\exp\left[\sum_{k,d} z_{ik}x_{id}\log\Big\{\frac{\theta_{kd}}{1-\theta_{kd}}\Big\}\right] \qquad (13)$$

This leads to :

$$\begin{aligned} p(x_i|z_i,\Theta) &= \frac{1}{c}\exp\left[\sum_{k,d}\log\Big\{\Big(\frac{\theta_{kd}}{1-\theta_{kd}}\Big)^{z_{ik}x_{id}}\Big\}\right] \qquad (14) \\ &= \frac{1}{c}\prod_{k,d}\left[\frac{\theta_{kd}}{1-\theta_{kd}}\right]^{z_{ik}x_{id}} \qquad (15) \end{aligned}$$

Now,

$$\begin{aligned} p(x_{id}|z_i,\Theta) &= \sum_{x_{i,-d}} p(x_{id},x_{i,-d}|z_i,\Theta) \qquad (16) \\ &= \sum_{x_{i,-d}}\frac{1}{c}\prod_{k,d'}\left[\frac{\theta_{kd'}}{1-\theta_{kd'}}\right]^{z_{ik}x_{id'}} \qquad (17) \\ &= \prod_k\left[\frac{\theta_{kd}}{1-\theta_{kd}}\right]^{z_{ik}x_{id}}\sum_{x_{i,-d}}\frac{1}{c}\prod_{k,d'\neq d}\left[\frac{\theta_{kd'}}{1-\theta_{kd'}}\right]^{z_{ik}x_{id'}} \qquad (18) \end{aligned}$$

Denote

$$\lambda = \sum_{x_{i,-d}}\frac{1}{c}\prod_{k,d'\neq d}\left[\frac{\theta_{kd'}}{1-\theta_{kd'}}\right]^{z_{ik}x_{id'}} \qquad (19)$$

Then,

$$p(x_{id}=1|z_i,\Theta) = \lambda\prod_k\left[\frac{\theta_{kd}}{1-\theta_{kd}}\right]^{z_{ik}} \quad \text{and} \quad p(x_{id}=0|z_i,\Theta) = \lambda \qquad (20)$$

Therefore, $x_{id} \sim \text{Bernoulli}(\tilde{\Theta}_d)$, with parameter :

$$\tilde{\Theta}_d = \frac{\prod_k \theta_{kd}^{z_{ik}}}{\prod_k(1-\theta_{kd})^{z_{ik}} + \prod_k \theta_{kd}^{z_{ik}}} \qquad (21)$$

where d is the dimension of x_i.

The two previous example illustrate that it is simple to derive the form of the product of the densities when we model the clusters to be part of exponential families. Thus, we do not need to dive into heavy calculus to compute the normalizing constant c.

In our implementation, we used Gaussian and multivariate Bernoulli clusters.

4 Infinite Overlapping Mixture Models via the IBP

The key point of the previous section is the use of vector z_i which defines which clusters point i belongs to : given z_i, we are able to construct a distribution for overlapping clusters.

Though, we don't necessarily know the clusters assignment nor the number of clusters. We thus have to make assumptions on the distributions of the vectors z_i. In this section, we derive a distribution over these binary assignment vectors z_i.

Denote π_k the probability of belonging to cluster k. Then $\mathbf{P}(z_{ik} = 1|\pi_k) = \pi_k$, or equivalently [6] :

$$z_{ik}|\pi_k \sim \text{Bernoulli}(\pi_k) \tag{22}$$

As we do not have any knowledge on the π_k, we turn to a Bayesian approach by giving each π_k a distribution. For simplicity, we would like this distribution to be conjugate to the Bernoulli, so we choose a Beta distribution [7] :

$$\pi_k|\alpha \sim \text{Beta}(\frac{\alpha}{K}, 1) \tag{23}$$

That is, $z_{ik}|\pi_k$ follows a Beta distribution.

As we can see below, decreasing the ratio $\frac{\alpha}{K}$ (by increasing K for instance) decreases the prior probability of belonging to a given cluster.

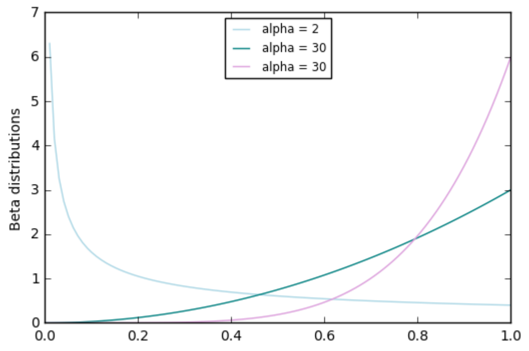


FIGURE 1 – Beta distributions for different values of the parameter α

The number K of clusters being also unknown, we wish to allow K to reach infinity. In the next part, we will define a distribution over the binary assignments z, resulting in a process known as the Indian Buffet Process (IBP) [8].

4.1 The idea behind the Indian Buffet Process

In their article, Griffiths and Ghahramani construct a probability distribution over equivalence classes [9] of binary matrices with a finite number of rows and an unbounded number of columns. This distribution

6. Remind that we don't need the restriction $\sum_k \pi_k = 1$ as the clusters are not mutually exclusive.
7. See proof in appendices.
8. Griffiths and Ghahramani. Infinite Latent Feature Models and the Indian Buffet Process. 2005.
9. See appendices for further explanations.

is suitable for use as a prior in probabilistic models that represent objects using a potentially infinite number of features, in our cases the cluster assignments.

The idea is to define a simple distribution that is exchangeable over the objects (rows). A simple choice is to assume that the columns are independent. Each column k is therefore modeled through parameter π_k (which can be thought as the frequency of feature k), and as we previously said, the full specification of the model requires a prior distribution $\pi_k|\alpha \sim \text{Beta}(\frac{\alpha}{K}, 1)$.

Let Z be a binary matrix of size $(N \times K)$, where N is the number of data and K the number of clusters, the i^{th} row being vector $z_i = [z_{i1}, ..., z_{iK}]$.

As we are interested in infinitely many possible features, we have to examine the limit $K \to \infty$. With our choice of parameters for the prior, the limit on the distribution of Z has nice properties :

— The number of ones in each row is distributed as Poisson(α) ;
— The total expected number of ones is αN ;
— The number of non-zero columns grows as $O(\alpha \log N)$;
— The rows are exchangeable.

This distribution is the IBP.

To understand how the IBP works, Griffiths and Ghahramani take the example of a set of customer queuing in an Indian buffet restaurant. More specifically, they define the distribution over infinite binary matrices by specifying how customers (objects) choose dishes (features).

Consider N to be the number of customers, and K the number of dishes. Each row of the matrix Z corresponds to a customer, and $z_{ik} = 1$ if customer i got dish k, 0 otherwise.

The IBP, which results in the same distribution over equivalent classes [10], works as follow :

— The first customer starts at the left of the buffet and takes a serving from each dish, stopping after a Poisson(α) number of dishes.
— The i^{th} customer moves along the buffet, sampling dishes in proportion to their popularity, taking dish k with probability $\frac{m_k}{i}$, where m_k is the number of previous customers who have sampled that dish. Having reached the end of all previous sampled dishes, the i^{th} customer then tries a Poisson$(\frac{\alpha}{i})$ number of new dishes.

As we already said, the number of features possessed by each individual follows a Poisson(α) distribution.

4.2 Computing the Z matrix

Going back to our problem, we can make use of computational algorithms to do inference on models using the IBP. We wish to compute matrix Z defined above.

Choosing $z_{ik}|\pi_k \sim \text{Bernouilli}(\pi_k)$ and $\pi_k|\alpha \sim \text{Beta}(\frac{\alpha}{K}, 1)$, we can compute the posterior distribution.

We have :

$$\begin{aligned} \mathbf{P}(Z|\pi) &= \prod_{k=1}^{K} \prod_{i=1}^{N} \mathbf{P}(z_{ik}|\pi_k) && (24) \\ &= \prod_{k=1}^{K} \pi_k^{\sum_{i=1}^{N} z_{ik}} (1-\pi_k)^{N-\sum_{i=1}^{N} z_{ik}} && (25) \\ &= \prod_{k=1}^{K} \pi_k^{m_k} (1-\pi_k)^{N-m_k} && (26) \end{aligned}$$

10. We invite the reader to report to the annexes. In this paper, we chose to present the Indian Buffet Process (which results in the same distribution) as it is a simpler and more convivial approach.

where $m_k = \sum_{i=1}^N z_{ik}$ is the number of points belonging to cluster k.

Integrating out π, we obtain the probability of Z :

$$\mathbf{P}(Z) = \prod_{k=1}^{K} \frac{\frac{\alpha}{K}\Gamma(m_k + \frac{\alpha}{K})\Gamma(N - m_k + 1)}{\Gamma(N + 1 + \frac{\alpha}{K})} \tag{27}$$

One may want to use a Gibbs sampler algorithm and calculate at each step $\mathbf{P}(z_{ik} = 1|\mathbf{Z}_{-(ik)}, X)$, which can be written as follow :

$$\mathbf{P}(z_{ik} = 1|\mathbf{Z}_{-(ik)}, X) \propto \mathbf{P}(X|\mathbf{Z})\mathbf{P}(z_{ik} = 1|\mathbf{Z}_{-(ik)}) \tag{28}$$

The last term can be computed [11] as :

$$\begin{aligned}
\mathbf{P}(z_{ik} = 1|\mathbf{z}_{-i,k}) &= \mathbf{E}[\mathbf{1}_{\{z_{ik}|z_{-ik}\}}] && (29)\\
&= \mathbf{E}_{\pi_k}\Big[\mathbf{E}[\mathbf{1}_{\{z_{ik}|z_{-ik}\}}|\pi_k]|z_{-ik}\Big] && (30)\\
&= \int_0^1 \mathbf{P}(z_{ik}|\pi_k)\mathbf{P}(\pi_k|\mathbf{z}_{-i,k})d\pi_k && (31)\\
&= \frac{m_{-i,k} + \frac{\alpha}{K}}{N + \frac{\alpha}{K}} && (32)
\end{aligned}$$

with $m_{-i,k}$ denoting the number of points belonging to cluster k, not including i.

Now, we take $K \longrightarrow \infty$ and get [12] :

$$\mathbf{P}(z_{ik} = 1|\mathbf{z}_{-i,k}) = \frac{m_{-i,k}}{N} \tag{33}$$

The advantage of the IBP is that it possess the property of exchangeability. That is, we can always imagine that the row we are sampling corresponds to the last customer to have entered the buffet.

5 IOMM Learning

We laid the foundations of the model, that we now combine together. To do so, we can include the IBP prior over the assignment vectors into the Overlapping Mixture Model that has been defined before, to obtain the Infinite Overlapping Mixture Model.

In the algorithm, we wish to :
— Determine the number of clusters K
— Set the parameters of each cluster
— Assign each data point to the clusters, i.e. determine the matrix Z

For a data point i, we denote k_+ the number of clusters which other data points belong to. Given the matrix Θ, the IBP matrix Z is sampled element-by-element. Algorithm 2 gives the general procedure :

11. Details of the calculus can be found in : T.L. Griffiths and Z. Ghahramani The Indian Buffet Process : An Introduction and Review, Journal of Machine Learning Research, pp. 1185–1224, 2011.
12. Only for k such that $m_{-i,k} > 0$

```
Initialize parameters $\Theta$, number of clusters $K$ and matrix $Z$;
for $j = 1$ to $n\_iter$ do
    for $i = 1$ to $N$ do
        for $k = 1$ to $k_+$ do
            $z_{ik} \sim z_{ik}|z_{-ik}, x_i, \Theta$ where $p(z_{ik} = 1|z_{-ik}, x_i, \Theta) \propto \frac{m_{-ik}}{N} p(x_i|\Theta, z_{ik} = 1, z_{-ik})$
        end
        Propose adding new clusters to point $i$, by drawing the number of new clusters from a
          Poisson($\frac{\alpha}{N}$)
        Draw parameters for the new clusters from the prior
    end
    Resample $\Theta|Z, X$ using MH proposal
end
```

Algorithm 1: IOMM Learning algorithm

We now explain with further details algorithm 2.

We first initialize the number of clusters K (which we will set at $K = 1$, as the algorithm does not allow to remove clusters), the matrix Z (where values are set as 0, except for the first column where values are set to 1 with probability 0.5). We also initialize the set of parameters Θ (for example, with binary data and multivariate Bernoulli clusters, we draw the parameters from a $\mathcal{U}[0, 1]$).

When adding new cluster assignments to a point i, we use a method developed by (Meeds et al., 2007) [13]. More specifically, instead of proposing only the number of new clusters, we jointly propose the number of new clusters along with the new parameters within a Metropolis-Hastings algorithm.

The acceptance ratio is then

$$r = \frac{P(X|n_B^*, \theta^*)}{P(X|n_B, \theta)} \tag{34}$$

where n_B^* is the proposed number of new cluster, drawn from a Poisson($\frac{\alpha}{N}$) with parameter θ^*, and θ^* is drawn from the prior given n_B^*.

This slightly modified version of the Metropolis-Hasting algorithm is motivated by the fact that we usually assume a conjugate situation in which the parameters Θ are explicitly integrated out of the model to compute the marginal likelihood $P(X|n_B^*)$, which is not the case here.

Finally, another Metropolis-Hastings is implemented in order to resample $\Theta|Z, X$ at each iteration (as the model is non-conjugate). We propose θ'_{kd}, where d is the dimension of our data points. The proposal distribution is $T(\theta_{kd}|\theta'_{kd}, \omega)$, with ω a control parameter.

The acceptance ratio is

$$a = \frac{p(\theta'_d|Z, x_d)}{p(\theta_d|Z, x_d)} \frac{T(\theta_{kd}|\theta'_{kd}, \omega)}{T(\theta'_{kd}|\theta_{kd}, \omega)} = \frac{p(x_d|Z, \theta'_d)p(\theta')}{p(x_d|Z, \theta_d)p(\theta)} \frac{T(\theta_{kd}|\theta'_{kd}, \omega)}{T(\theta'_{kd}|\theta_{kd}, \omega)} \tag{35}$$

where $p(\theta)$ is a prior over parameter θ.

6 Performances on toy datasets

We now generate simple data and examine the performances of the IOMM. We generated 3 datasets to examine the performance of the model when :
— Clusters do not overlap (mutually exclusive clusters) ;
— All clusters overlap ;
— Clusters may overlap or not.

13. E. Meeds, Z. Ghahramani, S. Roweis, and R. Neal. Modeling Dyadic Data with Binary Latent Factors. In *NIPS*, 2007.

In each of these 3 examples, we generated 100 binary data according to multivariate Bernoulli clusters.

Recall that the columns in the Z matrix can be permuted, we can not directly compare our learned matrix to the true matrix Z. To get rid of this issue, we compute at each iteration a matrix $U = ZZ^T$ of size $N \times N$ which we compare to the true matrix U^*.

Element (i, j) of this matrix is given by $U_{(i,j)} = \sum_{k=1}^{K} z_{ik} z_{jk}$. We see that $z_{ik} z_{jk} = 1$ only if data points i and j belong to cluster k. Therefore, $U_{(i,j)}$ gives the number of shared clusters between i and j.

As the matrix Z is unstable, our predicted $\hat{U}$ matrix is obtained by averaging the U matrices we get at each iteration. That is, we ran 5000 iterations, burning in the first 1000.

We then compute the following statistics to evaluate the performance of the model :

$$|(\hat{U} - U^*)| = 0 \tag{36}$$

$$|(\hat{U} - U^*)| \leq 1 \tag{37}$$

$$|(\hat{U} - U^*)| \leq 2 \tag{38}$$

The first one returns the number of pairs of data points that share the same clusters as in $\hat{U^*}$, whereas the second and the third return the number of pairs that differ by at most one and two clusters respectively.

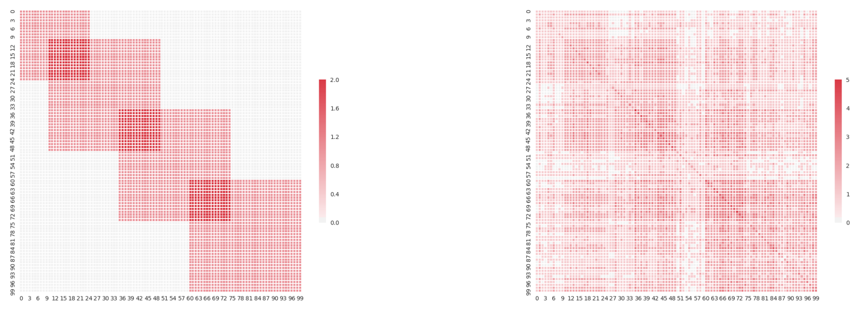
6.1 Binary data

First, we generated 100 two-dimensional binary data according to 2 "naive" clusters which do not overlap. We run $3,000$ iterations with a burn-in of $2,000$. The model performs well as it succeeds to reproduce the true matrix U, as shown by the figures in the appendices, which proves that the implementation works well.

Then, we generated 100 fifty-dimensional binary data according to 5 non-overlapping clusters, where the parameters of each cluster were drawn uniformly between 0 and 1. We run $9,000$ iterations, burning-in the first $5,000$. There again, the results are satisfying as the model succeeds to discover all the clusters, as shown in the appendices.

6.2 Gaussian clusters

Next, we tested the model on a more complex dataset, generating data according to 4 overlapping Gaussian clusters. As we did previously, we run $9,000$ iterations, burning-in the first $5,000$ iterations. The model struggles a bit more to reproduce true matrix U but results are overall satisfactory.



(a) True U (b) Simulated U

Figure 2 – Model performances on 4 Gaussian overlapping clusters

As we can see, the model finds too many clusters, which also explains the form of the simulated matrix $\hat{U}$ we have.

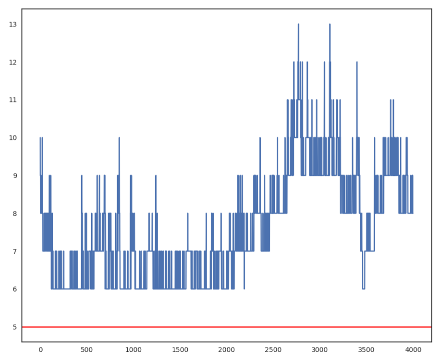


FIGURE 3 – MCMC iterations for the 4 overlapping Gaussian clusters

Though, the number of pairs of data points that differ by at most one cluster represents 84% of the set.

Statistic	Percent
$\lvert(\hat{U} - U^*)\rvert = 0$	24.00 %
$\lvert(\hat{U} - U^*)\rvert \leq 1$	84.00 %
$\lvert(\hat{U} - U^*)\rvert \leq 2$	99.00 %
$\lvert(\hat{U} - U^*)\rvert \leq 3$	100.00 %

TABLE 1 – Statistics for learned matrix $\hat{U}$ in the case of Gaussian clusters

7 Applying overlapping clustering methods to gene expression data

7.1 Motivations

Gene expression data are generated from experiments with high-throughput technologies, such as microarrays and RNA-Seq. Usually, thousands of genes are investigated by measuring their expression levels under different experimental conditions, such as tissue samples (e.g., normal and cancerous tissues) or time series during a biological process. These data are generally organized as a matrix, where each row represents a gene, each column corresponds to an experimental condition, and each cell stands for the expression level of a gene under this specific condition.

Data clustering techniques became very popular tools for gene expression data analysis, being used for different purposes, such as functional annotation or tissue classification. In these problems, researchers typically apply a partitional or hierarchical clustering algorithm which uses a (dis)similarity measure that takes into account all gene or condition dimensions. Although useful, these approaches suffer from serious drawbacks. First of all, each gene or experimental condition is assigned to only one cluster. Besides, each gene or experimental condition must be assigned to some cluster. These assumptions may not reflect the reality, since a gene or experimental condition could take part of several biological pathways, which in turn could be active only under subsets of experimental conditions or genes.

In order to overcome these limitations, it becomes necessary to apply techniques capable of identifying subsets of genes with similar behaviors under subsets of experimental conditions. Such techniques should also allow a gene or experimental condition to take part of one or more clusters. From this perspective, the main objective is to simultaneously cluster rows and columns of a data matrix in order to find homogeneous submatrices, which can possibly overlap. Each of these submatrices is called a bicluster, and the process of finding them is called biclustering (or coclustering) in literature.

The previously explained *Nonparametric Bayesian Approach to Modeling Overlapping Clusters* thus perfectly fits to this problematic, as evoked by the authors in the abstract. As of our knowledge, this algorithm has not yet been applied to gene expression data. In order to compare our results to previous literature, we implemented another biclustering algorithm, detailed in the next section.

7.2 An alternative approach : the Plaid biclustering algorithm

One of the first who introduced a basic idea of biclustering to the research community was Hartigan [1972]. His approach is an example of a divide-and-conquer method, which aims to minimize the overall variance within a cluster by splitting a given partition into two clusters. The split which maximizes the sum of squares reduction is chosen. The splitting of the matrix is stopped when a further splitting reduces the sum of squares less than expected by chance [14]. Pioneers in applying bicluster algorithms to gene expression data were Cheng and Church [2000]. The basic idea behind their greedy iterative search approach called the δ-method is, that there are specific row, column and submatrix effects within a bicluster. Thus, each bicluster has got a residual score

$$H(I,J) = \frac{1}{|I||J|} \sum_{i \in I, j \in J} (a_{ij} - a_{iJ} - a_{Ij} + a_{IJ})^2,$$

where a_{iJ} is the mean of row i (row effect), a_{Ij} is the mean of column j (column effect) and a_{IJ} is the overall mean of the bicluster (submatrix effect). If the score of a certain submatrix A_{IJ} is below a threshold δ the submatrix is called a bicluster.

The algorithm starts from the entire matrix A and deletes rows and columns with the highest score as long as $H(I,J) > \delta$. In a second step rows and columns with the lowest score are being added as long as $H(I,J) < \delta$. The described process is repeated until an appropriate number of biclusters is found.
The particular biclustering algorithm we developed in order to compare the results of the bayesian non-parametric approach is called the Plaid algorithm, and is one of the most efficient and widely used in literature. The idea behind the plaid model is that a data matrix can be described as a linear function of layers. That means, the expression level of a matrix is the sum of K biclusters and a uniform background noise term. To specify, the matrix entries can be written as

$$a_{ij} = \sum_{k=0}^{K} \theta_{ijk} \rho_{ik} \kappa_{jk}$$

where $\rho_{ik} \in \{0,1\}$ and $\kappa_{jk} \in \{0,1\}$ are indicator variables indicating whether a gene i or sample j belongs to the k-th bicluster or not. Overlapping / exhaustiveness criteria can be precised by setting constraints on these indicators. θ_{ijk}, which specifies the contribution of each element to the matrix, can be defined very flexibly. In our case, we will assume a so-called Plaid linear model in which $\theta_{ijk} = \mu_k + \alpha_{ik} + \beta_{kj}$, where μ_k is the background noise term in bicluster k, and α_{ik} and β_{kj} respectively describe row and column specific effects in bicluster k. Noting μ_0 the background noise term of the whole matrix, the plaid model can eventually be written as

$$a_{ij} = \mu_0 + \sum_{k=1}^{K} (\mu_k + \alpha_{ik} + \beta_{kj}) \rho_{ik} \kappa_{jk}$$

Noting n the number of genes and p the number of samples considered, the problem can naively be seen as finding the vector $\widehat{\theta} = (\theta_1, \ldots, \theta_K)$ minimizing the residual sum of squares :

$$\widehat{\theta} = \text{argmin} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{p} (a_{ij} - \theta_{ij0} - \sum_{k=1}^{K} \theta_{ijk} \rho_{ik} \kappa_{jk})^2$$

The research of new biclusters is done step-by-step. Assuming we are looking for an N^{th} layer given that the N-1 first ones have already be found, the optimization program is to minimize

$$Q = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{p} (R_{ij}^{N-1} - \theta_{ijN} \rho_{iN} \kappa_{jN})^2$$

14. See Appendice for details

where R_{ij}^{N-1} is the residual from the N-1 first layers. At each of these steps, we determine θ, ρ and κ thanks to an iterative approach[15].

The last parameter to fit in Plaid algorithm is the number of biclusters K that have to be found. Unlike the bayesian non-parametric approach, K has to be limited by an upper-bound in order that the algorithm stops. This is done by giving ourselves a maximum number of biclusters K_{max} and until this number, we compare new layers found to noise. Noting σ_k^2 the sum of squares of layer k, we want to compare it to the sum of squares generated by simple noise. We model noise by randomly permuting every row and every column of the residual matrix obtained after extraction of K-1 first layers, and this a certain number R of times, and we note $\widetilde{\sigma}^{2,r}$ its residual sum of squares. The acceptance rule for the layer k is then to accept it if and only if

$$\sigma_k^2 > \underset{1 \leq r \leq R}{\widetilde{\sigma}^{2,r}}$$

Choosing large values of R increases computational costs (linear in R) but in the same time increases the significance of our results as it decreases the probability of accepting a layer that is only background noise.

7.3 Dataset used

The data used for implementation of both algorithms consists in a subsample of the Saccharomyces Cerevisiae organism, which is a type of yeast organism, i.e. eukaryotic, single-celled microorganisms. This dataset, called BicatYeast, is a widely used dataset to test biclustering algorithms in literature and was first used by Barkow et al. (2006) to present biclustering. More specifically, the dataset is microarray data of 419 genes over 70 experiments with information about the expression level regarding these two dimensions. The values in the matrix are previously log transformed. As a consequence, a negative value in position (i, j) means that the gene i under condition j is under-expressed compared to a reference sample, and symmetrically a positive value means it is over-expressed.
Regarding the Plaid method, biclustering directly looks for submatrixes of the microarray data, choosing 'individuals' and 'variables' thus has no impact on the algorithm. However regarding the Bayesian overlapping clustering approach, we decided to use the experiments as our 'individuals' that we allow to belong to more than one cluster composed of several genes.

7.4 Results

In this section, we present the results obtained on the dataset regarding the two approaches previously presented.

7.4.1 Plaid algorithm

The Plaid algorithm was ran with an upper bound of 50 maximum number of biclusters. 12 different biclusters were found with an effective overlapping regarding genes for 5 clusters and regarding experiments (conditions) for 6 clusters. The dimensions of the clusters found are presented in the table below. The number of the bicluster corresponds to its statistical significance regarding the algorithm presented in the previous section.

Bicluster number	Number of genes in bicluster	Number of experiments in bicluster
1	75	5
2	38	3
3	20	6
4	142	6
5	34	4
6	25	3
7	111	3
8	85	8
9	125	2
10	4	5
11	8	8
12	28	2

TABLE 2 – Dimensions of biclusters found

15. See Appendice for details

Interestingly enough, the two most important clusters found correspond respectively to an under-expression and over-expression of genes on the subset of experiments, as we can see in the two graphs plotted below.

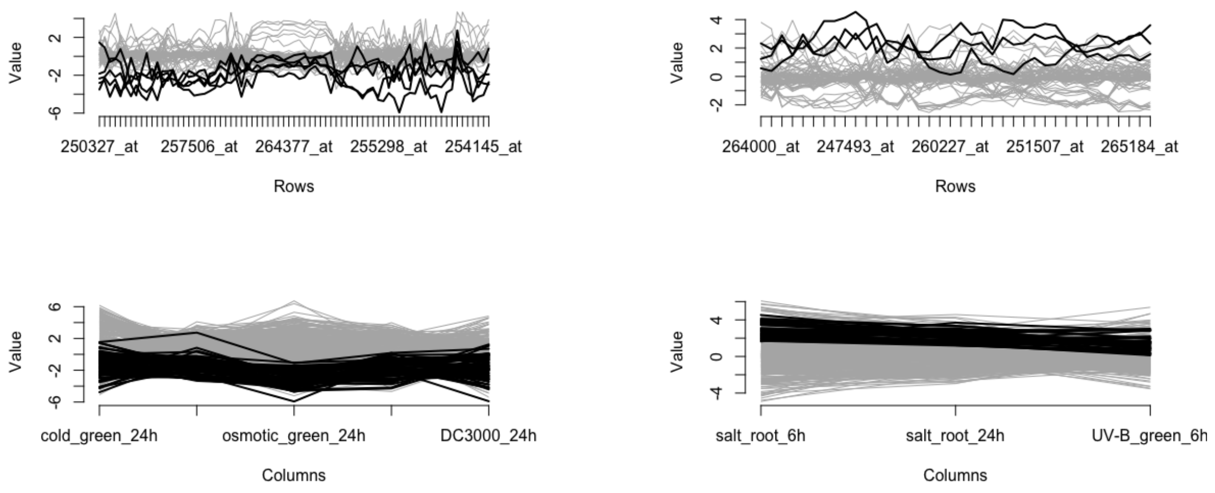


FIGURE 4 – First two biclusters expression levels found by Plaid algorithm

On total, 208 genes were involved in at least one cluster. When involved in several clusters, the mean number of clusters implicated was of 2.35. Two genes (*245343_ at* and *245688_ at*) were involved in 10 of the 12 discovered clusters by the Plaid algorithm, which shows to what extent it is important to allow overlapping clusters in both dimensions.
Regarding experiments (i.e. conditions), 13 of the total 70 were involved in at least one cluster.

7.4.2 IOMM

We propose to use the IOMM algorithm to cluster the experiments. By using the estimated parameters for the clusters we can describe the composition of each clusters.

The algorithm finds 4 clusters, one of them containing a majority of the data points. The learning took at least 8 hours. Too few clusters were accepted. One of the possible reasons is the fact that we did not use a normal inverse Wishart proposal when re-sampling Θ. However, using a normal inverse Wishart proposal would have significantly increased the computational time. Instead, we proposed the μ parameters according to a $\mathcal{U}[-1, 1]$, and the σ parameters according to a $\mathcal{U}[0, 1]$. Our choice was motivated by the good results we got on the toy dataset.

We think that one of the reasons the algorithm struggles to discover new clusters is the number of variables. Randomly selecting acceptable parameters over 70 dimensions is unlikely to happen. Thus, the algorithm should need more iterations or an increased α. But in both cases, the training time would be unbearable for us.

8 Conclusion

In this paper, we implemented a model that allows to cluster data belonging to several clusters. The model showed good results on simple toy datasets, but struggled to discover the right number of clusters when we tested it on overlapping Gaussian clusters.

One of the drawbacks of the model is that it is computationally very costly. Training time is linear with the number of individuals and with the number of dimensions. More over, the more dimensions there are, the more iterations are needed to find the relevant clusters as the new clusters are always proposed independently of the previous found clusters. Thus, when applying it on a real-world dataset with only

419 individuals and 70 variables, it took several hours to perform only $3,000$ iterations. We don't think it is an implementation issue because in the original paper, the authors also experimented their model on a small dataset.

Parallelization may be a way to tackle this issue, but it has to be done in a cautious way regarding the structure of the algorithm. The re-sampling of Θ could be parallelized as each component of the parameter is proposed and accepted independently. In contrast, it appears that the Gibbs sampler part cannot be parallelized easily because of the way the IBP proposes and fills new clusters. However, one could possibly parallelize the Gibbs sampler by segmenting the dataset into several parts, independently treat them, and then aggregate them. This would greatly degrade the performance of each individual iteration but the speed-up of the algorithm might allow to perform way more iterations and achieve greater overall performances.

9 References

[1] C. Bishop. *Pattern Recognition and Machine Learning*, Chapter 9. Springer, 2006.

[2] Y. Chen and M. Church. Biclustering of Expression Data. Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology, ISMB'00, 93-103, 2000.

[3] T. Griffiths and Z. Ghahramani. Infinite Latent Feature Models and the Indian Buffet Process. Technical report, Gatsby CNU, 2005.

[4] J. A. Hartigan. Direct Clustering of Data Matrix. Journal of the American Statistical Association, Vol. 67, No.337, pp. 123-129, 1972.

[5] T. Griffiths and Z. Ghahramani The Indian Buffet Process : An Introduction and Review, Journal of Machine Learning Research, pp. 1185–1224, 2011.

[6] K. A. Heller, Z. Ghahramani. A Nonparametric Bayesian Approach to Modeling Overlapping Clusters. Journal of Machine Learning Research 2 :187-194, January 2007.

[7] E. Meeds, Z. Ghahramani, S. Roweis, and R. Neal. Modeling Dyadic Data with Binary Latent Factors. In *NIPS*, 2007.

[8] L. Lazzeroni, and A. Owen. Plaid models for gene expression data. Statistica sinica, pp. 61-86, 2002.

[9] B. Pontes, R. Giráldez and J.S. Aguilar-Ruiz. Biclustering on expression data : A review. Journal of biomedical informatics, 57, pp. 163-180, 2015.

10 Appendices

10.1 Proof of Beta conjugate prior to Bernoulli likelihood

We have :

- $z_{ik}|\pi_k \sim$ Bernouilli(π_k)

- $\pi_k|\alpha \sim$ Beta$(\frac{\alpha}{K}, 1)$, i.e. $p(\pi_k) \propto \pi_k^{\frac{\alpha}{K}-1}$

The posterior is give, by :

$$\begin{aligned} p(\pi_k|z_{ik}) &\propto p(z_{ik}|\pi_k)p(\pi_k) && (39)\\ &\propto \pi_k^{z_{ik}}(1-\pi_k)^{1-z_{ik}}\pi_k^{\frac{\alpha}{K}-1} && (40)\\ &\propto \pi_k^{z_{ik}+\frac{\alpha}{K}-1}(1-\pi_k)^{1-z_{ik}} && (41)\end{aligned}$$

Thus,

$$\pi_k|z_{ik} \sim \text{Beta}(z_{ik} + \frac{\alpha}{K}, 2 - z_{ik}) \tag{42}$$

10.2 Equivalence of binary matrices

We define the equivalence classes of binary matrices, which can be viewed as the analogue of partitions for class assignments.

The equivalence classes is defined by Griffiths and Ghahramani with respect to the $lof(.)$ function, which is a function on binary matrices that maps binary matrices to left-ordered binary matrices.

Let Z be a binary matrix. Then, $lof(Z)$ is obtained by ordering the columns of the binary matrix Z from left to right by the magnitude of the binary number expressed by that column, taking the first row as the most significant bit.

For example, to obtain the left-ordered binary matrix of Z, the columns for which $z_{1k} = 1$ are grouped at the left. In the second row, the columns for which $z_{2k} = 1$ are grouped at the left of the sets for which $z_{1k} = 1$, and so on.

We say that two binary matrices Y and Z are lof-equivalent if $lof(Y) = lof(Z)$, and we denote by $[Z]$ the equivalence class of a binary matrix Z. This equivalence class is defined as the set of all binary matrices that are lof-equivalent to binary matrix Z.

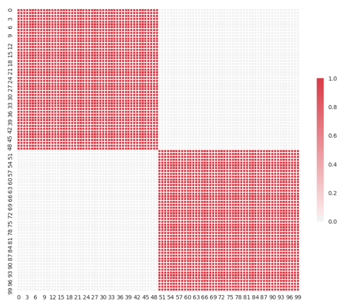
The advantage of lof-equivalent classes is that they preserve permutation of rows and columns of a matrix.

Cardinality of lof-equivalence class $[Z]$ is given by :

$$\binom{K}{K_0 ... K_{2^N-1}} = \frac{K!}{\prod_{h=0}^{2^N-1} K_h!} \tag{43}$$

where K_h is the number of columns with full history h ; the history of feature k at object i being defined as $(z_{1k}, ..., z_{(i-1)k})$.

10.3 Results on the toy datasets


(a) True U

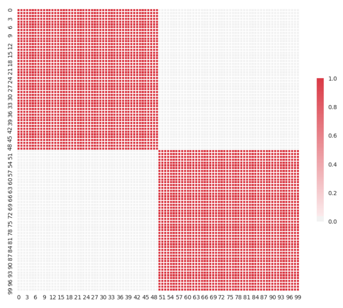

(b) Simulated U

FIGURE 5 – Model performances on binary with 2 non-overlapping clusters

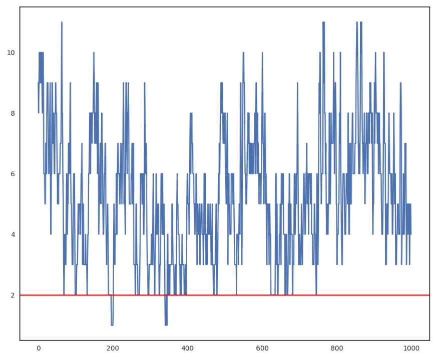
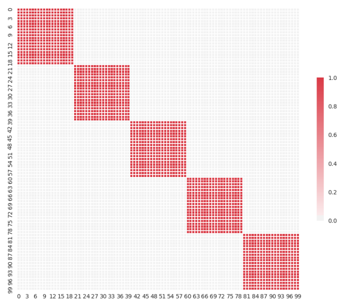


FIGURE 6 – MCMC iterations for the 2 non-overlapping clusters


(a) True U

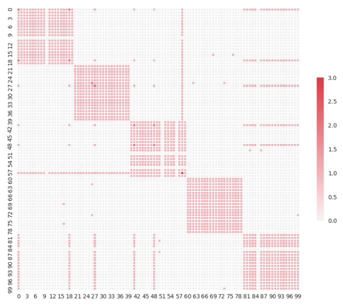

(b) Simulated U

FIGURE 7 – Model performances on binary data with 5 non-overlapping clusters

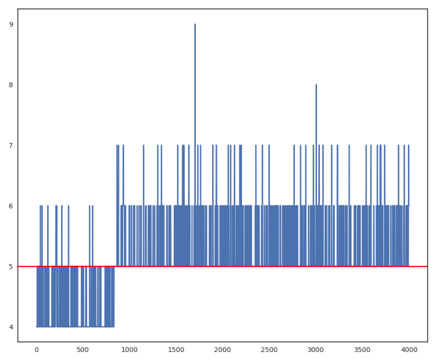


FIGURE 8 – MCMC iterations for the 5 non-overlapping clusters

10.4 Plaid iterative algorithm

```
Require : K-1 first layers;
for $s = 0$ do
    Determine $\rho^{(0)}, \kappa^{(0)}$ by power-iteration [a]
end
for $s = 1$ to $S$ do
    Update $\theta$ : $\theta^{(s)} \leftarrow \underset{\theta}{\mathrm{argmin}} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{p} (R_{ij} - \theta_{ij} \rho_i^{(s-1)} \kappa_j^{(s-1)})^2$
    Update $\rho$ : $\rho^{(s)} \leftarrow \underset{\rho}{\mathrm{argmin}} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{p} (R_{ij} - \theta_{ij}^{(s)} \rho_i \kappa_j^{(s-1)})^2$
    Update $\kappa$ : $\kappa^{(s)} \leftarrow \underset{\kappa}{\mathrm{argmin}} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{p} (R_{ij} - \theta_{ij}^{(s)} \rho_i^{(s)} \kappa_j)^2$
    if $\rho_i^{(s)} < 0.5$
    $\rho_i^{(s)} \leftarrow \rho_i^{(s)} + \frac{s}{2S}$
    else if $\rho_i^{(s)} > 0.5$
    $\rho_i^{(s)} \leftarrow \rho_i^{(s)} - \frac{s}{2S}$
    if $\kappa_j^{(s)} < 0.5$
    $\kappa_j^{(s)} \leftarrow \kappa_j^{(s)} + \frac{s}{2S}$
    else if $\kappa_j^{(s)} > 0.5$
    $\kappa_j^{(s)} \leftarrow \kappa_j^{(s)} - \frac{s}{2S}$
end
```

Algorithm 2: Find optimal parameters of a given layer K

a. Power iteration is a widely used algorithm for finding the most important eigenvalue and its associated eigenvector of a sparse matrix