Chapter 2
Step 1: Clustering Data

2.1 Introduction

2.1.1 Clustering

Clustering reduces the size of the data by replacing individual genes with artificial super-genes that can be treated as single nodes for the purposes of network inference. By clustering genes that work together as a preprocessing step, we can improve the accuracy of the resulting network by reducing variance due to noise on individual genes. The goal is to generate clusters while losing the minimum amount of information in the dataset (and perhaps even make certain relationships stronger!). For example, if there are two genes that both behave in exactly the same way across the experimental conditions of interest, then little to no information is lost if you treat them as though they were a single "gene".

There are many different clustering algorithms available. The basic idea in all clustering algorithms is to group data that are alike together based on some measure of similarity. A classic example is the K-Means algorithm. K-Means takes a parameter k that specifies the number of clusters to generate. In the random start method, K "centroids" are chosen at random (e.g. these may be the expression profiles of k genes). Then (step 1) each expression profile is compared with all the centroids and is assigned to the most similar one. The expression profiles corresponding to a given centroid are called a cluster. Next (step 2), for each cluster a centroid that better represents the expression profiles of that cluster is chosen. Steps 1 and 2 are repeated until there is no further change in cluster membership. Then the whole procedure begins again with a new set of initial centroids. After trying something like 100 different sets of centroids, the algorithm starts with the best clustering found, where best means that each centroid is closest on the average to all the members. [1]

[1] Other schemes such as kmeans++ have also been proposed [1], but we like the random starting point approach for its sheer simplicity.

J. M. Lingeman and D. Shasha, *Network Inference in Molecular Biology,*
SpringerBriefs in Electrical and Computer Engineering,
DOI: 10.1007/978-1-4614-3113-8_2, © The Author(s) 2012

2.1.2 Biclustering

Biclustering is a method of clustering across both rows and columns, i.e. across gene expressions and conditions. Biclustering begins with a matrix where each column corresponds to an experiment and each row is a gene. Traditional clustering attempts to shuffle the rows into groups based on their expression values. However, this works poorly when grouping genes by expression values for different experiments, because genes may behave quite differently across experimental conditions, as we have mentioned above. Biclustering solves this by allowing the shuffling of both genes and experiments. This allows clusters to be selected based on a subset of experimental conditions rather than all of the experimental conditions in a given gene's row.

Genes may end up in more than one bicluster. Biologically, this makes sense, as a single gene may interact with different sets of genes in different conditions. For example, gene X may be part of cluster Y under condition A, but part of cluster Z under condition B.

2.2 cMonkey

2.2.1 What it Does

Finding the best biclustering (according to virtually any reasonable criterion) is NP-complete. cMonkey [26] combines biological intuition with heuristics to come to high quality biclusters in all cases we have tested.

2.2.2 The Data

The data used can be broken up into three types: expression data, upstream DNA sequence data, and (where available) experimentally verified network connections. The expression data are steady-state data, though time-series data can be used as well.

2.2.3 The Strategy

The algorithm starts by seeding biclusters with genes that are co-expressed across most if not all conditions (this is the basic strategy, though there are variants). After that, a gene or condition can be associated with a bicluster based on a score that is formulated as a probability of association with the current members of that biclus-

ter. A gene can be associated with many biclusters depending on whether its score exceeds a certain threshold (using several different criteria). As the membership of biclusters changes, a particular gene may move in or out of a particular bicluster.

The criteria to determine whether a gene and/or experiment should belong with a bicluster are expression similarity, promoter similarity, and known network connections. We first show how to calculate the score that determines whether a gene or experimental condition should be associated with a bicluster based on expression (2.2.3.1). Second we describe the calculation of a score based on DNA motifs (small binding portions of a promoter) to determine whether the promoter of a gene warrants that gene's association with a bicluster (2.2.3.2). Third we describe how to use known network edges (e.g. metabolic edges) to determine whether a gene should be associated with a bicluster (2.2.3.3). Finally, we show how all these different criteria can be combined using a heuristic technique called simulated annealing into a single regression score (2.2.3.4 and Equation (2.4)).

The basic idea is that at each step of the algorithm, we calculate the probability that each gene g and each experiment e belongs in bicluster k, given the genes and experiments already in k. Based on that probability, the gene-experiment may be added to the bicluster. Biclusters are initialized by randomly choosing one of several methods (discussed in section 2.2.3.4). The probability of movement decreases over time (in the spirit of simulated annealing). The algorithm converges to a set of stable biclusters. The probability that a gene or condition should belong to a given bicluster is calculated for each of the types of data used at each iteration. These probabilities are then combined using a regression model, giving an overall probability that a given gene belongs to each bicluster.

We present the details of the algorithm below as four discrete steps. First, we calculate a likelihood score for each type of data separately. Then, we combine the scores into one value. Finally, we go over cMonkey's iterative procedure, putting all of the previous steps together. The net result is the likelihood a gene or condition/experiment should belong to a given bicluster.

2.2.3.1 Using the Expression Data

Expression data are used to create a likelihood that a given gene or experiment "belongs" to a given bicluster. We calculate the likelihood that x_{ij}, gene i's expression value in experiment j, is in a bicluster k with:

$$p(x_{ij}) = \frac{1}{\sqrt{2\pi\left(\sigma_j^2 + \varepsilon^2\right)}} \exp\left(-\frac{1}{2}\frac{\left(x_{ij} - \bar{x}_{jk}\right)^2 + \varepsilon^2}{\sigma_j^2 + \varepsilon^2}\right), \tag{2.1}$$

where σ_j^2 is the variance of experiment j, ε is an error term, representing unknown error in the expression values, and $\bar{x}_{jk}$ is the mean expression level of experiment j over the genes in bicluster k. The variance over all of j is used instead of the variance of only the genes in bicluster k. This is done to help weed out experiments

where there is not much variation between genes, i.e., experiments where genes are more likely to be correlated by random chance.

Once we have the likelihood that each measurement x_{ij} belongs to each bicluster k, we can calculate the likelihood that each gene and experiment belongs to a given bicluster. To calculate the likelihood that a gene i belongs to a bicluster, we take the product of the likelihoods that gene i is in bicluster k across all experiments j.

$$\text{prob gene } i \text{ belongs to bicluster } k = \prod_{j \in J_k} p(x_{ij}). \tag{2.2}$$

Similarly, we calculate the likelihood that an experiment j belongs to a bicluster by taking the product of the likelihoods that experiment j is in bicluster k across all genes i.

$$\text{prob condition } i \text{ belongs to bicluster } k = \prod_{i \in I_k} p(x_{ij}). \tag{2.3}$$

cMonkey then assigns a gene to a bicluster over the conditions of that bicluster if the probability of that gene belonging to a bicluster is greater than a randomly generated value between 0 and 1. Finally, we create a co-expression p-value for each gene i with respect to each bicluster k and for each experiment j with respect to each bicluster k, labeled r_{ik} and r_{ij}, respectively. These values are created by integrating over the normal distribution based on (2.1).

2.2.3.2 Using the Upstream DNA Sequence Data

Upstream DNA sequence data are used to identify whether or not a gene i shares DNA motifs with other genes in a bicluster k. A DNA motif is a sequence of DNA bases that are likely to be a target of transcription factors. So if two genes share the same motifs then they may have similar biological functions. The MEME algorithm [2] is used to identify these motifs, and the counterpart algorithm MAST [2] is used to calculate the p-value that a given sequence matches motifs found by MEME. MEME is used to create a set of the motifs found in the genes in each bicluster. MAST is then used to calculate a p-value s_{ik} of how likely it is that a gene i contains the motifs found in bicluster k. Intuitively, what we are checking is whether or not a gene shares similar biological functions (based on their DNA) with the rest of the genes in a given bicluster.

2.2.3.3 Using the Association Network Data

cMonkey also uses known association network data to help create biclusters. The network data can be obtained from the KEGG [16], Predictome [23] and Prolinks [3] databases. These datasets contain known network associations between genes. The basic idea here is to add genes to a cluster based on how many network associations that gene has in common with the genes in that cluster. The more associations a

gene has in common with a cluster, the more likely it is that the gene belongs in that cluster. A p-value q_{ik}^n is calculated for each gene/cluster/network set, where n is each one of the different types of association networks (e.g., KEGG or Prolinks).

2.2.3.4 Putting it all together: The cMonkey Iterative Procedure

cMonkey starts by "seeding" each bicluster with an initial set of genes and experiments. Five different methods are used for seeding the initial biclusters:

1: A single random gene
2: Using co-expressed genes from another clustering method
3: Using semi-co-expressed genes (by correlation of expression values)
4: Using highly connected genes (from the association network)
5: Using genes with a common motif (from the upstream DNA sequence)

Many seeding strategies are used in order to introduce variance into the initialization of the algorithm. This helps keep the algorithm from getting caught in a local minimum early on.

cMonkey uses simulated annealing to select the biclusters. Simulated annealing is a probabilistic global optimization algorithm where a temperature parameter starts "hot" and "cools" over time. When the temperature is hot, genes and experiments are allowed to move much more freely between biclusters. This allows many combinations to be formed early on. As the temperature cools, moves between clusters become less and less likely, based on how "good" the move is. A gene that has a high affinity with a bicluster, as then constituted, has a higher probably of moving to it than one that doesn't. As the temperature cools, the biclusters "harden", until only the genes that are extremely good matches for a bicluster have any chance of actually moving to that bicluster. The maximum number of moves that can be made at each iteration is set by a parameter to be a small value, by default 5. Establishing a maximum is done to ensure that a bicluster cannot change too much in a single iteration.

In addition to the temperature parameter used in simulated annealing, the weights on the importance of each data type also change at each iteration. For example, Reiss et al. state that early in the procedure, DNA motifs are not very useful, as it is unlikely that any particular cluster has enough similar motifs to give a reliable signal. However, some of the association network data can be extremely informative early on. Thus, at the beginning of the algorithm, association network data are given a higher weight than the motif data. As the annealing continues and clusters become more and more numerous, using DNA motifs makes more sense, so the weight on DNA motifs is increased over time.

A constrained logistic regression is used to combine each data type's score into a single joint likelihood. First, the scores are standardized to have mean 0 and standard deviation 1, with $log(\tilde{z_{ik}}) = log(z_{ik}) - \mu_k/\sigma_k$ (where z is a stand-in for the expression (r), sequence (s), or association network q^n p-value). This is done so that one type of score doesn't overpower the others simply because it tends to have better p-values

than the others. Then, we combine the scores into a single value, on which we will perform the regression:

$$g_{ik} = r_0 \log(\tilde{r}_{ik}) + s_0 \log(\tilde{s}_{ik}) + \sum_{n \in N} q_0^n \log(\tilde{q}_{ik}^n), \tag{2.4}$$

where r_0, s_0, and q_0^n are the weights that are specified by the current annealing iteration for the expression, sequence, and network scores, respectively. The idea here is to weight each type of p-value based on how important we think it is given where in the annealing process we are.

The constrained logistic regression is then defined as:

$$\pi_{lk} \equiv p(y_{lk} = 1 | X_k, S_i, M_k, N) \propto \exp(\beta_0 + \beta_1 g_{lk}), \tag{2.5}$$

where l is used to define a gene *or* experiment (replacing i and j from before). π_{ik} is then the likelihood that a given gene or experiment l belongs to bicluster k.

cMonkey puts all of these methods together into a single iterative procedure. (i) Create random biclusters using a random initialization method for each bicluster, and start at a high annealing temperature. (ii) Calculate the joint likelihood that each gene/experiment belongs in each bicluster. (iii) For each gene / experiment add or drop it from each bicluster according to the probabilities:

$$p(\text{add}|\pi_{lk}) = e^{-\pi_{lk}/T}; p(\text{drop}|\pi_{lk}) = e^{-(1-\pi_{lk})/T}, \tag{2.6}$$

where T is the current annealing temperature or until the maximum number of moves per iteration is achieved (the default value is 5) and π_{lk} is the likelihood that a given gene or experiment l belongs to bicluster k, as defined in Equation (2.5). (iv) Lower the annealing temperature, then repeat steps (ii) - (iv) until the temperature reaches 0 or its minimum value. By default, the annealing temperature begins at 0.15 and goes down in even steps to 0.05 over 100-150 iterations.

The result of this procedure yields biclusters of genes and experiments. These data can be used to create "super genes" for use in a network inference algorithm. By taking the list of genes in a bicluster, we can then average those values together to create the expression values over experiments for this gene. The idea behind this is that if these genes are part of the same pathway and behave like each other, then we can reduce the amount of noise and variance in the expression measurements by averaging their values together.

2.2.4 Walkthrough Example on Toy Data

We will now demonstrate cMonkey in action using a small toy example. Our example will contain only 5 genes and 5 experiments. We begin by randomly selecting one of the five different methods for seeding a bicluster. Let's assume that we choose method 1: selecting a single random gene to start a bicluster and begin with

an annealing temperature of 0.15. Gene 3 is selected to begin the bicluster. We then calculate the joint likelihood that each gene or condition belongs in this bicluster. Table 2.1 represents the joint probability (the result of Equation (2.4)) that each gene/experiment belongs in a bicluster containing only gene 3.

	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5
Gene 1	0.8	0.2	0.01	0.3	0.02
Gene 2	0.3	0.9	0.1	0.1	0.6
Gene 4	0.5	0.6	0.4	0.3	0.2
Gene 5	0.2	0.1	0.7	0.2	0.3

Table 2.1 Score of each gene/experiment with respect to gene 3.

We can see from Table 2.1 that there are some gene/experiment combinations that have a good chance of actually belonging to this bicluster. Specifically, Gene 1/Experiment 1 and Gene 2/Experiment 2. Given the joint probabilities in Table 2.1, we calculate the probability of each gene/experiment being added to the bicluster using the left side of Equation (2.6). This takes into account the current annealing temperature, which is set to 0.15 at the beginning. This gives us Table 2.2, which shows 1 – score.

	Experiment 1	Experiment 2	Experiment 3	Experiment 4	Experiment 5
Gene 1	0.99	0.73	0.07	0.86	0.12
Gene 2	0.86	0.99	0.49	0.49	0.98
Gene 4	0.97	0.98	0.93	0.865	0.74
Gene 5	0.74	0.49	0.99	0.73	0.86

Table 2.2 1 – Score of each gene/experiment with respect to the new gene 3 bicluster. The numbers are high because the annealing temperature is high. As the temperature cools, these scores will decrease.

Gene/experiment values with high joint probabilities in Table 2.1 have correspondingly high numbers in Table 2.2. To decide whether or not to add a particular gene/experiment gx to the bicluster, we select a random number between 0 and 1. If that random number is less than the value shown in Table 1.2 for gx, then we add gx. Otherwise, don't add gx to the bicluster. For example, suppose we select a random number of 0.6 for the value of gene 1/experiment 1. Because the random number is less than the listed value, we add gene 1/experiment 1 to the bicluster. By contrast, if, for gene 1/experiment 2, we select a random number of 0.9. We do not add that gene/experiment combination to the bicluster.

A similar process is then carried out to decide whether or not to drop gene/experiment pairs from the bicluster, using the right side of Equation 2.6 instead of the left. Once this process has been carried out for each bicluster (in this example, there is only one), the annealing temperature is dropped. Dropping the annealing temperature af-

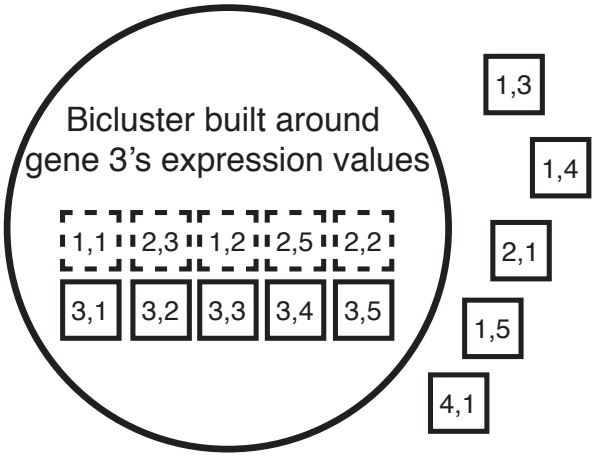

Fig. 2.1 A figure showing an example bicluster that was seeded with the expression values of gene 3. The observations are labeled as boxes, with the numbers representing "gene number, experiment number". Dashed boxes are the 5 freshly added observations, and solid boxes indicate observations that were already part of the bicluster. Shown outside of the bicluster are some other observations that did not make the cut.

fects the results of Equation 2.6, making it less likely that a move is made. This process continues until either the minimum annealing temperature is reached or until a maximum number of moves (additions or deletions) is made for this turn, yielding a collection of biclusters.

Parameter Name	What it does	Default Value
m	Maximum number of moves per iteration	5
$T_{\max}$	The starting annealing temperature	0.15
$T_{\min}$	The ending annealing temperature	0.05
T_{step}	The number of steps between $T_{\max}$ and $T_{\min}$	100

2.3 Factor Analysis for Bicluster Acquisition (FABIA)

2.3.1 What it Does

Factor Analysis for Bicluster Acquisition (FABIA) [14] biclusters genes and experiments using Factor Analysis. Factor Analysis takes a set of data (in our case, the expression values of genes in experiments) and explains them in terms of an often smaller set of parameters called factors. In a non-genetic context, consider lung

cancer data about people. Imagine that the data are rows of people with lung cancer and columns are indirect data that are correlated with a direct cause of the cancer (age, socio-economic status, location, etc.). What FABIA tries to do is explain the underlying relationship between indirect data. For example, a direct cause of lung cancer such as asbestos exposure (i.e., a "factor") may be partially explained by a combination of other features correlated with the direct cause, such as whether or not a person lived in a time and area where asbestos exposure was common. FABIA uses Expectation-Maximization [6] to generate the biclusters. Biclusters are then ranked by mutual-information content, and weaker members of each bicluster are optionally pruned with a threshold.

2.3.2 The Data

FABIA uses steady-state data. The experiments can be genetic or external perturbations, and should be normalized to have mean 0 and standard deviation 1. Time-series experiments may also be used, but they will be treated as individual steady-state experiments.

2.3.3 The Strategy

FABIA tries to find a set of factors z that explain observed expression values in X. To do this, we need to find a good set of weights called "factor loadings" that connect a factor in z to the observation in X. Which genes are part of a given factor z_i is decided by the factor loadings in λ associated with z_i. We also want to model the measurement noise ε of each observation and then remove that noise in order to calculate an "idealized" expression value (see Figure 2.2).

Formally, this can be modeled as:

$$X = \sum_{i=1}^{p} \lambda_i \tilde{z}_i + \varepsilon = \Lambda \tilde{z} + \varepsilon, \tag{2.7}$$

where ε is additive noise, p is the number of biclusters, λ_i is a sparse vector of factor loadings, and $\tilde{z}_i$ is the i^{th} value in a vector of $\tilde{z}$ factors. The approach to fitting this model uses some advanced techniques.

For this model, we want to find the parameters Λ and Ψ that best explain the data. $\Psi = \text{Cov}(\varepsilon)$ is a matrix that represents the covariance of the noise of the expression values in X. Λ (the factor loadings) represents the connections between factors in z and observations in X. We find parameters to best explain X using the Expectation-Maximization (EM) [6] algorithm.

Expectation-maximization (EM) is an iterative method for finding the maximum likelihood of a set of parameters. FABIA uses a special kind of EM algorithm group

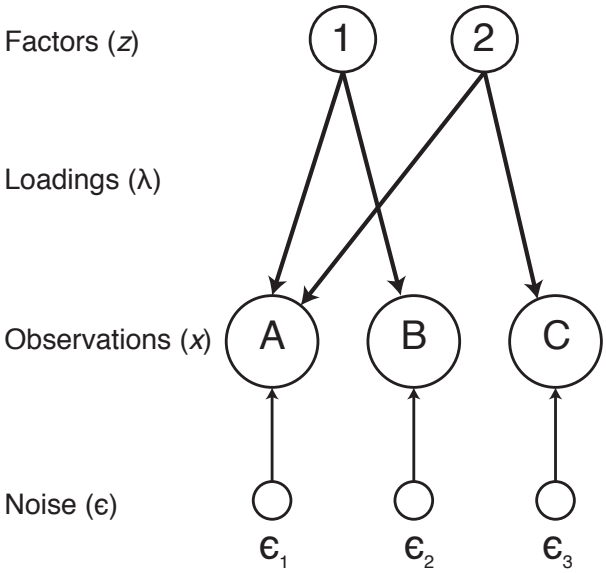

Fig. 2.2 A figure showing the relationships between the different elements of FABIA. In this case, the two factors (z) explain the three observations (x) through factor loadings (λ). Factor 1 connects observations A and B. Factor 2 connects observations A and C. FABIA treats these factors as biclusters, giving us 2 biclusters where bicluster 1 contains A and B, and bicluster 2 contains A and C.

called "variational EM" [10] [24]. This implementation of the variational EM algorithm places a Laplacian prior on the problem to enforce sparsity. This is similar to the constraint that l_1 optimization uses: non-zero values are implicitly penalized, so "weak" connections will quickly drop to zero, yielding a more parsimonious result. For variational EM, this is done by optimizing for the variational parameters ξ that are part of the Lagrangian dual formulation of the problem. By optimizing the variational parameters, we obtain a lower bound on the likelihood of the model. The goal is to find a maximum value for this lower bound. FABIA uses variational EM to search for the combination of Λ and Ψ that fits the data the best. This works in two steps. First, the expectation (E) step calculates the expected log-likelihood of the current parameters, i.e., how likely it is that the current parameters fit the data better than a null model. Next, the maximization (M) step computes a new set of model parameters Λ and Ψ that maximize the expected log-likelihood from the previous E-step. These two steps are repeated until a maximum number of iterations is reached or the model is changing too slowly.

Once we've obtained a good estimate of the parameters Λ and Ψ, we can rank the resulting biclusters according to how much information each contains about the data. We do this by calculating the mutual information between the data X and the factors of each bicluster z_i^T. The information content of a bicluster will grow with the number of nonzero values (i.e., the size of the bicluster) in each λ_i, so in general,

the larger the biclusters are then the more information about X they contain. Finally, each bicluster may optionally be pruned by taking the absolute value of the factors and factor loadings for each bicluster, and selecting only the values that are above a certain threshold.

2.3.4 Walkthrough Example on Toy Data

FABIA works by iteratively adjusting the loading matrix Λ and the covariance matrix of the noise Ψ. Imagine that we are working with a very small dataset of only 9 observations (3 genes, 3 experiments). We are looking to fit these 9 observations into 3 factors (biclusters). Λ is then a 3x9 matrix initialized randomly to values between 0 and 1. Ψ is initialized to be:

$$\Psi = \text{diag}(\text{CoV}(x) - \Lambda\Lambda^t). \tag{2.8}$$

The variational parameters ξ are initialized to be 1.

Our initial guess, a randomly generated matrix, is a mess. It is likely that it is fully connected, i.e., there is an edge from each observation to each bicluster. This is obviously not a useful result, so we want to begin pruning the edges that don't belong. We then enter the E-step of the EM algorithm, where we calculate the log-likelihood that the current parameters Λ and Ψ fit the data better than a null model. The likelihood that each observation x_i is a factor z_j is calculated. Obviously, at this point that, likelihood is going to be very low, as we have a random model right now.

Next, we use this likelihood in the M-step in order to find new matrices Λ and Ψ that better explain the model. To update Λ, a convex quadratic problem is solved. The basic idea though is that we use the likelihood from the E-step that each observation in X belongs to a given factor in z, given the rest of the observations currently in that factor. The factor loading in Λ^{new} is then updated to reflect the likelihood that an observation x_i is in factor z_j. Small values of factor loadings are penalized and forced to 0. Then, Ψ is updated using the updated Λ^{new}. The basic idea behind updating Ψ is to use the new information available from Λ^{new} and from the likelihoods calculated during the E-step to estimate the variance in the observations X. If an observation strongly belongs into a bicluster, then maybe that gene really does have a very high or very low value, and it isn't simply a noisy observation. In this case, the value in Ψ is reduced.

At the end of the M-step we have a slightly better guess as to which genes actually belong to each bicluster. For our example, let's say that Λ now looks like Table 2.3. We can see that observation 8 seems to strongly belong to factor 1, and that observation 5 strongly belongs to factor 2. There are also several small values. These are likely to drop to 0 in following iterations as their small values are penalized. As more EM iterations are performed, these numbers will slowly change until either the numbers converge to values and stop changing or we hit the maximum number

	Obs. 1	Obs. 2	Obs. 3	Obs. 4	Obs. 5	Obs. 6	Obs. 7	Obs. 8	Obs. 9
Factor 1	0	0.5	0.3	0.1	0	0.2	0.9	0.4	0.2
Factor 2	0.1	0	0	0.9	0.5	0	0.2	0.1	0.6
Factor 3	0.2	0.1	0	0.4	0.3	0.1	0	0.5	0.3

Table 2.3 Example Λ matrix after 1 Expectation-Maximization step.

of iterations. The matrix Λ is then a map between our factors and observations, and this map is used to create the biclusters.

Parameter Name	What it does	Default Value
p	Maximum number of biclusters	Depends on data
cyc	Maximum number of iterations	500