Flexible model-based co-clustering – FlexiCoClustering manual

Joint clustering of binary and continuous data

Authors: Fatin Zainul Abidin*, David Westhead**

Contacts:

*bs12fnza@leeds.ac.uk

**D.R.Westhead@leeds.ac.uk

Section 1: Introduction to algorithm

The method is designed to cluster multiple data of different types where each entity for clustering is described by a sets of binary and continuous variables. It is generically applicable to a range of different problems in biology. It uses a simple model based framework based on a joint probability distribution over binary and continuous variables that is a mixture over a variable number of clusters. It uses penalized maximum likelihood (ML) estimation of mixture model parameters using information criteria and meta-heuristic searching for optimum clusters by Monte-Carlo simulated annealing (SA). The program takes as input a mixture of binary data (e.g. presence/absence of mutations, motifs, regulatory input, epigenetic marks etc.) and continuous data (e.g. gene expression, protein abundance, metabolite levels) for a list of samples (e.g. genes, patients). This program works best with smaller and concise datasets, thus pre-filtered data to only important features is preferable. To date, this program works well with ~1000 rows in the input file (number of entities to cluster) and longer run time might be needed for larger datasets to converge to a good solution. An example of pre-filtering of a dataset would be reducing the number of genes to only highly variable genes. Upon taking the input files required, the program will run until either the termination or convergence criterion are met. The clustering solution is then refined using expectation maximization, taking the simulated annealing solution as the starting point.

Platform Dependencies

Task Type: Clustering of data points

CPU Type: any

Operating System: any

Language: Java - JDK 1.8

Section 2: How to run the Program

? How to run the FlexiCoClustering using command-line interface (FlexiCoClustering-CLI)

1. Download all the files from https://github.com/BioToolsLeeds/FlexiCoClusteringPackage/
2. To run the demo , use the following command:

   java -jar <path to the FlexiCoClustering.jar/FlexiCoClustering.jar> <Input.txt> <Output.txt>

   Press enter
3. To re-submit the same job with restart after program termination:

   Change Nrun: '0' to Nrun: '1'

   Increase the MaxTemps to a higher value than the previous run if the MaxTemps iteration was

   Completed or else just use default MaxTemps parameter. Then, use the following command:

   java -jar <path to the FlexiCoClustering.jar/FlexiCoClustering.jar> <Input.txt> <Output.txt>

   Press enter
4. The program can be terminated at any time by pressing Ctrl+C.

A detailed description of the runtime parameters and input file format is given in section 3 below.

? How to run the FlexiCoClustering using GUI based (FlexiCoClustering-GUI)

1. Download all the files from https://github.com/BioToolsLeeds/FlexiCoClusteringPackage/
2. To run the demo, use the following command:

   java -jar <path to the FlexiCoClustering.jar/FlexiCoClustering.jar> or double click the .jar file

3. A graphical user interphase (GUI) window will be opened and looks like this snapshot below:

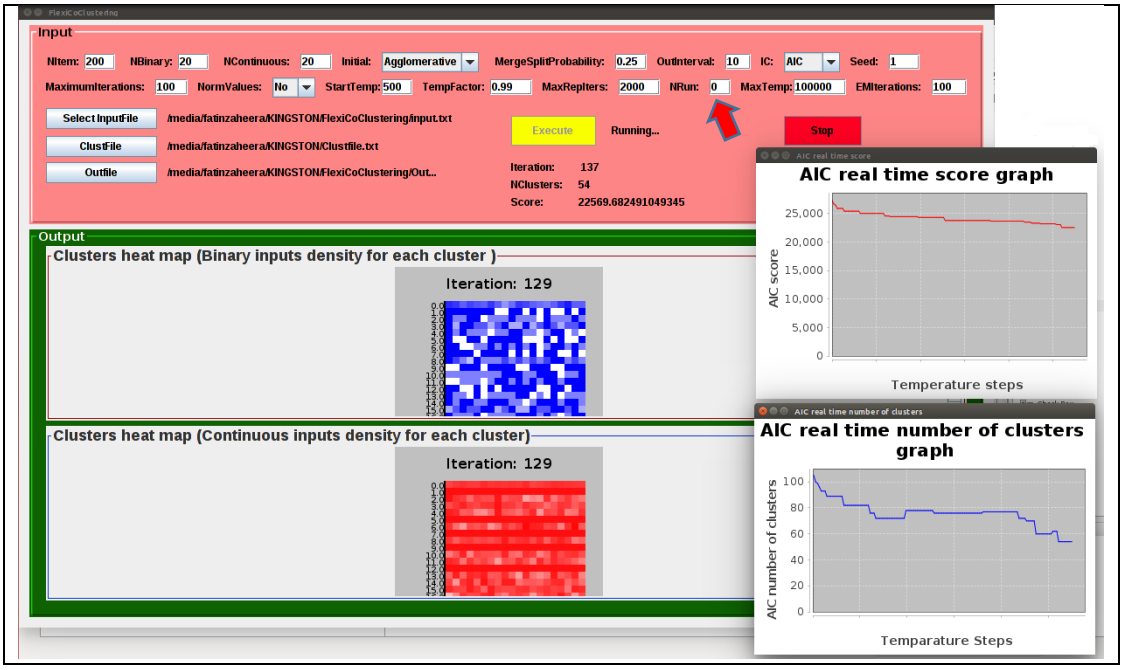

Figure 1: A snapshot of the FlexiCoClustering GUI upon submitting the java -jar command on the terminal/command prompt. Red arrow shows where user should change the NRun to '1' after the initial run (NRun=0) have finished if it is required at all.

4. On the GUI options (using demo example):

"ClustFile" : Name the ClustFile as i.e. Clustfile.txt
"Outfile" : Name an Outfile as i.e. Output.txt
"Select inputFile" : Select <path to the FlexiCoClustering.jar>\example\input.txt

For the real run using user own data, please change parameters accordingly. Parameters are described in **table 2** on the next page.
Press the "Execute" button. On default, this program will run for 100000 temperature steps (MaxTemps) and produce two real-time updated heat map image files (.png) in every 10 iterations interval for binary and continuous variables. The Output.txt and Clustfile.txt will be updated as the program progress and once the program terminated respectively.
An EM refining file (EMRefinement.txt) containing all the marginal densities for the clusters will be produced automatically in the same working directory.
If more than initially specified number of maximum temperature steps (MaxTemps) is required after step 2 had finished, re-run the program by first by replacing '0' with '1' in the Nrun. Increase the MaxTemps to a higher value than the previous run if the MaxTemps iteration was completed or else just use default MaxTemps parameter.

Then, press the "Execute" button again. User can also terminate the run at any time by pressing 'Stop' button.

Section 3: Package input and runtime parameters

Input File

Input file (e.g. input.txt): A space separated formatted text file containing the binary and continuous input dataset and runtime parameters (command line interface only).

A.	NItems: 100 NBinary: 20 NContinuous: 20 Agglomerative IC: AIC MergeSplitProbability: 0.25 MaximumIterations: 100 StartTemp: 500 TempFactor: 0.999 MaxTemps: 5000 MaxRepIters: 2000 Seed: 1 EMIterations: 100 OutInterval: 10 ClustFile: clustfile.txt NormExp: 0 Nrun: 0 Gene 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 3.41 2.81 8.91 2.33 9.09 9.82 1.23 2.24 3.43 3.43 9.76 ... Gene 2 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 3.41 2.81 8.91 2.33 9.09 9.82 1.23 2.24 3.43 3.43 9.76 ... Gene 3 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 3.41 2.81 8.91 2.33 9.09 9.82 1.23 2.24 3.43 3.43 9.76 ... Gene 4 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 3.41 2.81 8.91 2.33 9.09 9.82 1.23 2.24 3.43 3.43 9.76 ... Gene 5 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 3.41 2.81 8.91 2.33 9.09 9.82 1.23 2.24 3.43 3.43 9.76 ...
B.	BinaryInputs: TF1 TF2 TF3 TF4 TF5 TF6 TF7 TF8 TF9 TF10 TF11 TF12 TF13 TF14 TF15 TF16 TF17 TF18 TF19 TF20 ContinuousInputs: Exp1 Exp2 Exp3 Exp4 Exp5 Exp6 Exp7 Exp8 Exp9 Exp10 Exp11 Exp12 Exp13 Exp14 Exp15 Exp16 ... Gene 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 3.41 2.81 8.91 2.33 9.09 9.82 1.23 2.24 3.43 3.43 9.76 ... Gene 2 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 3.41 2.81 8.91 2.33 9.09 9.82 1.23 2.24 3.43 3.43 9.76 ... Gene 3 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 3.41 2.81 8.91 2.33 9.09 9.82 1.23 2.24 3.43 3.43 9.76 ... Gene 4 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 3.41 2.81 8.91 2.33 9.09 9.82 1.23 2.24 3.43 3.43 9.76 ... Gene 5 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 3.41 2.81 8.91 2.33 9.09 9.82 1.23 2.24 3.43 3.43 9.76 ...

Table 1: Example of an input file for **A.** command line interphase (CLI) and **B.** graphical user interphase (GUI) based package. From 3rd or 21st row onwards of GUI or CLN based package respectively, first column shows the data points (i.e. gene names, sample names) and the second column onwards are the binary inputs ("1" and "0") followed by continuous values.

Name	Functional Description
Nitems	Number of data points to cluster. Must be equal to the number of data lines (rows) in the input file.
Nbinary	Number of binary variables per entity to be clustered. Must be equal to the number of 1/0s at the start of each data line in the input file.
NContinuous	Number of continuous variables in the input files. Must be equal to the number of floating point numbers at the end of each data line in the input file.
Agglomerative/Divisive	Starting point option for clustering, if agglomerative start with all data points in separate clusters, if divisive start with all in a single cluster.
IC (**AIC**)	Objective function/Information criterion (see table below for options available)
MergeSplitProbability (**0.25**)	Monte Carlo move: either an ordinary step (moving a data point between clusters) or a cluster merge/split according to this probability.
MaximumIterations (**100**)	Number of Monte-Carlo moves at each temperature
StartTemp (**500**)	Starting temperature of the simulated annealing. Higher temperature-more random solution will be accepted at the beginning of simulated annealing.
TempFactor (**0.99**)	Temperature reduction factor at each iteration of the temperature loop.
MaxTemps (**100000**)	Maximum number of temperature to be simulated (termination criterion). Higher value will make the SA runs longer.
MaxRepIters (**2000**)	Maximum number of simulated annealing best score repetitions. If the score does not change at up to this number of repetition, the SA will be terminated although the maximum temperature is not reached.
Seed (**1**)	The seed of the random number generator.
EMIterations (**100**)	Maximum number EM iterations
OutInterval (**10**)	Intervals at which the solution is printed to the output file and at which the heat maps are updated on GUI
ClustFile	The name of the final clusters output file
NormExp (**0**)	Normalizes continuous inputs to zero mean and a standard deviation for each data points- z-scores.
Nrun (**0**)	Number of re-run of the program after initial run

Table 2: Runtime parameters of both GUI and CLN based package. The underlined and bold values are the default values of the runtime parameters.

Output file (e.g. Output.txt)

A text file containing all the runtime updates such as score, current best modules/clusters, etc.

EM refinement file (e.g. EMRefinement.txt)

A text file containing all the marginal densities of each cluster found from the simulated annealing procedure.

Section 4: Mathematical representation of score calculation

Solution (clusters) score calculation:

$$p(r_{i1}, \dots, r_{in_r}, e_{i1}, \dots, e_{in_e}) = \sum_{m=1}^{N} \alpha_m \prod_{j=1}^{n_r} B(r_{ij}; p_{mj}) \prod_{l=1}^{n_e} N(e_{il}; \mu_{ml}, \sigma_{ml}) \quad \dots\dots\dots\dots\dots\dots\dots\dots \textbf{(1)}$$

N : data points *i*, representing genes, tumour samples etc.
$r_{ij} \in \{0,1\}, j = 1, \dots, n_r$ binary variables
$e_{il}, l = 1, \dots, n_e$ continuous variables
α_m are mixing coefficients $\sum \alpha_m = 1$.
B denotes the Bernoulli distribution with parameter p_{mj}, and *N* is a normal distribution with parameters μ_{ml} andσ_{ml}.
We assume a probability distribution (1) which is a mixture of N_m components (clusters).

In the case of genetic regulation the mixture components represent the well-known concept of a cluster of co-regulated genes, with, for example, Bernoulli parameters p_{mj} representing the probability of binding for particular transcription factors in promoter/enhancer elements, and the μ_{ml}representing a shared average pattern of gene expression, which could be a time or developmental series but is not required to be. In the case of tumour samples, clusters could be related samples where Bernoulli parameters associate mutation probabilities at particular loci with shared patterns of oncogenic gene expression.

Since the number of clusters in unknown and difficult to estimate, an initial heuristic search was adopted for an approximately optimal model, followed by refinement of the solution by expectation maximization. The heuristic search employed a Monte-Carlo simulated annealing algorithm (see **Algorithm 1**) to optimize objective functions of the form

$$\boldsymbol{O(L, k) = -2L + \lambda k(N)} \quad \dots\dots\dots\dots\dots\dots\dots \textbf{(2)}$$

where L is the (maximized) log-likelihood from the distribution above, λ is a function of the number of data points N and *k* is the number of parameters in the model.
Several different functions $\lambda(N)$ can be use with our algorithm as shown in table 3 below:

	Criterion	$\boldsymbol{\lambda}$	N	Equation	Reference
a.	AIC2	2	1	-2L + 2k	Akaike, 1973
b.	AIC2.5	2.5	1	-2L + 2.5k	-
c.	AIC3	3	1	-2L + 3k	Bozdogan, 1993
d.	HQC	2	ln ln(N_g)	-2L + 2k(ln(ln(N_g)))	Hannan and Quinn, 1979
e.	AIC4	4	1	-2L + 4k	-
f.	AIC5	5	1	-2L + 5k	-
g.	BIC	1	ln N_g	-2L + k(ln(N_g))	Scwarz,1978
h.	CAIC	1	ln N_g +1	-2L + k(ln(N_g)+1)	Bozdogan, 1987

Table 3: Different objective functions tested and can be chosen by user sorted ascendingly (from a. to h.) based on its stringency in penalizing free parameters and number of data points in the model.

Monte-Carlo simulated annealing (SA) for clusters optimization.

```
Normalize expression (genes)*
Clusters = Initialize clusters (agglomerative/divisive)
Best clusters = [list]
Score = 0.0
Old score = 0.0
Best score = 0.0
Difference = 0.0
Count temperature = 0
Temperature = Start temperature
While Count temperature < Maximum temperature:
        Temperature * Temperature decreasing factor
        Score = Calculate modules score (clusters)
        beta = 1.0/temp
        While Iteration < Maximum iterations: **
                If random  > Merge and split probability:
                        Change (clusters)
                Else:
                        If random > 0.5:
                                Merge (clusters)
                        Else: Split (clusters)
        Old Score = Calculate clusters score (clusters)
        Difference = Difference + (Score-Old score)
        If Difference < 0.0 or random < exponent(-beta*Difference):
                'accept'
                if Score < Best score:
                        Best score = Score
                        Best clusters = clusters
        Else:
                'reject'
                Score = Old score
        Count temperature + 1
        If Best score = Old Score:
                Count score = Count score + 1
        If Count score == 2000:
                break
return Best score, Best clusters
```

Algorithm 1: Pseudo-code for the Monte-Carlo Simulated annealing algorithm.

*Optional

**This step runs in parallel (minimum of 5 parallel threads).

Expectation-maximization

The parameters of model produced as the best solution from the heuristic search can be refined by EM, with the useful side effect of estimating the degree of mixing between modules through the probability density that data point *i* is generated from mixture component *m* (3).

$$p(m|i,\theta) = \frac{\alpha_m p_m(i|m,\theta)}{\sum_{j=1}^{N} \alpha_j p_j(i|j,\theta)} \quad \text{..... (3)}$$

This is derived from Bayes' rule: p_m is the probability density for mixture component *m* defined in 2.2, θ denotes the (current) vector of parameters for all modules and the mixing coefficients α_m can be interpreted as prior probabilities for membership of each module. The steps of the EM algorithm are showed in the figure as followed:

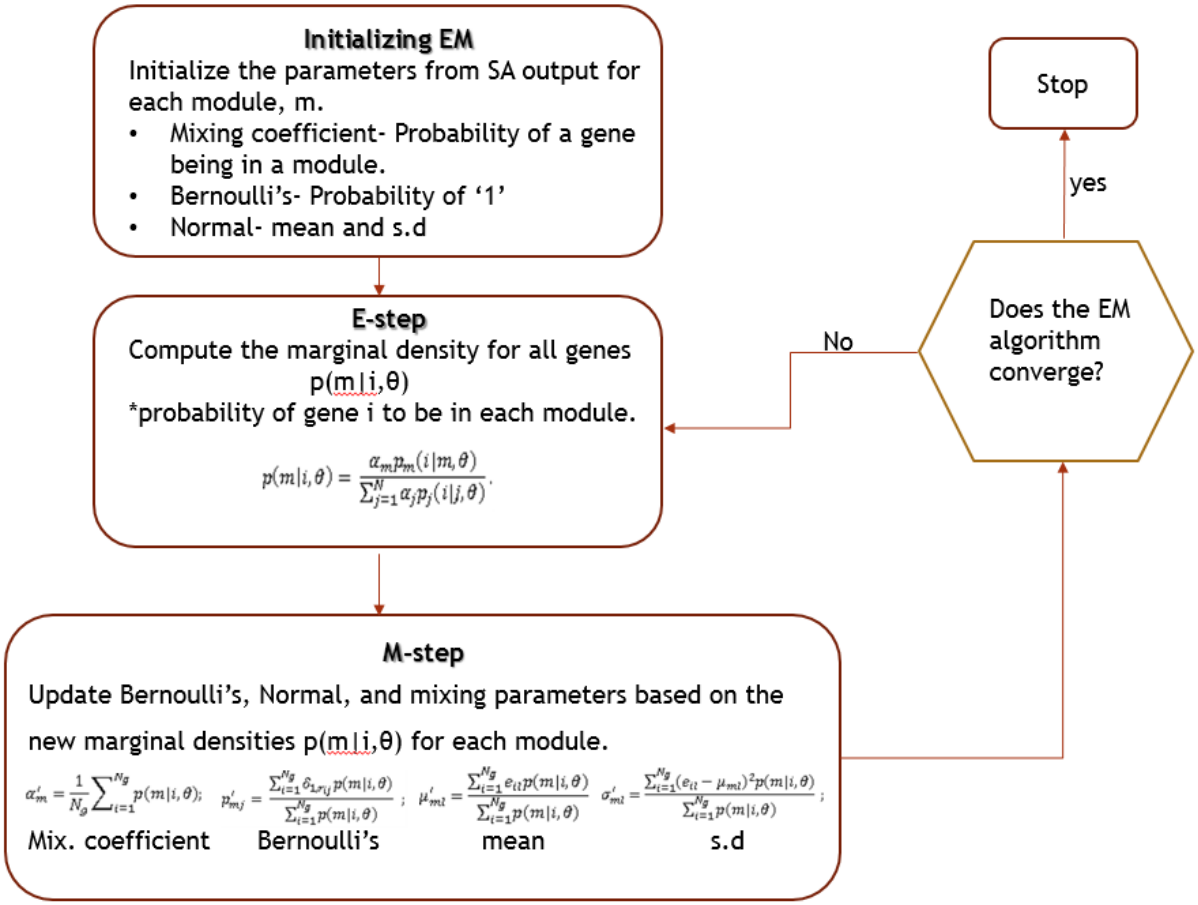

Figure 1: Refinement of model parameters using EM which starts with the prior probability densities from the SA output and refinement of its parameters until convergence. Convergence here means, until the parameters values do not changed for 2 consecutive EM iterations or until the maximum number of iterations has been reached.