A BAYESIAN APPROACH TO THE CLUSTERING PROBLEM WITH APPLICATION TO GENE EXPRESSION ANALYSIS

by

Işık Barış Fidaner

B.S., Computer Engineering, Bogazici University, 2005

M.S., Computer Engineering, Bogazici University, 2008

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2016

ACKNOWLEDGEMENTS

I thank my thesis supervisor Ali Taylan Cemgil, and our colleagues Betül Kırdar, Ayça Cankorur-Çetinkaya and Duygu Dikicioğlu.

ABSTRACT

A BAYESIAN APPROACH TO THE CLUSTERING PROBLEM WITH APPLICATION TO GENE EXPRESSION ANALYSIS

This thesis investigates methods for extraction of information from gene expression time series data. These time series provide indirect measurements about the underlying biological mechanisms, hence their analysis heavily depends on statistical modelling techniques. One particularly popular analysis approach is clustering genes by their similarity of expression profiles. However, for scientific data analysis, clustering requires a rigorous methodology and Bayesian nonparametrics provides a promising framework. In this context, two novel models were developed: Infinite Multiway Mixture (IMM) that extends the standard infinite mixture model; and Infinite Mixture of Piecewise Linear Sequences (IMPLS) that assumes a specific structure for its mixture components, tailored towards gene expression time series. In the Bayesian paradigm, the key object for gene analysis is the posterior distribution over partitionings, given the model and observed data. However, a posterior distribution over partitionings is a highly complicated object. Here, we apply Markov Chain Monte Carlo (MCMC) inference to obtain a sample from the posterior distribution of gene partitionings, and cluster genes by a heuristic algorithm. An alternative, novel approach for the analysis of distributions over partitions is also developed, that we named as entropy agglomeration (EA). We demonstrate the use of EA by a clustering experiment on a literary text, Ulysses by James Joyce. In our bioinformatics application CLUSTERnGO (CnG), the relevance of resulting clusters are evaluated by applying standard multiple hypothesis testing to compare them against previous biological knowledge encoded in terms of a Gene Ontology. The complete workflow of CnG consists of a four-phase pipeline (Configuration, Inference, Clustering, Evaluation).

ÖZET

ÖBEKLEME PROBLEMİNE BAYESCİ BİR YAKLAŞIM VE GEN İFADESİ ANALİZİNDE UYGULANMASI

Bu tezde gen ifadesi zaman serisi verisinden bilgi çıkarılması için yöntemler araştırılmıştır. Bu zaman serileri altta yatan biyolojik mekanizmalara dair dolaylı ölçümler sağlar, bu yüzden analizlerde istatistiksel modelleme tekniklerine yoğunca başvurulur. Özellikle popüler bir analiz yaklaşımı, ifade profili benzerliklerine göre genleri öbeklemektir. Fakat bilimsel veri analizi açısından öbekleme güçlü bir metodoloji gerektirir ve Bayesci nonparametri bu konuda gelecek vaat eden bir çerçeve sağlar. Bu bağlamda, iki yeni model geliştirildi: Standart sonsuz karışım modelini genişleten Sonsuz Çokyönlü Karışım (IMM); ve karışım bileşenlerinde gen ifadesi zaman serilerine uyarlanmış özgül bir yapıyı varsayım alan Parçalı Doğrusal Dizilerin Sonsuz Karışımı (IMPLS). Bayesci paradigmada gen analizi için anahtar nesne, model ve gözlemler verildiğinde, bölüntüler üzerindeki sonsal dağılımdır. Fakat, bölüntüler üzerinde bir sonsal dağılım oldukça karmaşık bir nesnedir. Burada Markov zinciri Monte Carlo çıkarımı uygulayarak gen bölüntülerinin sonsal dağılımından bir örneklem elde ediyoruz, ve sezgisel bir yöntemle genleri öbekliyoruz. Bölüntüler üzerindeki dağılımların analizi için entropi toplaşması (EA) adını verdiğimiz alternatif, yeni bir yaklaşım da geliştirildi. EA'nın kullanımı, edebi bir metne (Ulysses, James Joyce) uygulanan öbekleme deneyiyle gösterildi. Biyoenformatik uygulamamız olan CLUSTERnGO'da (CnG) sonuçta çıkan öbeklerin amaca uygunluğunu değerlendirmek için standart çoklu hipotez testi uygulanır, bir gen ontolojisine ait terimlerle kodlanmış önceki biyolojik bilgilerle karşılaştırılır. CnG'nin süreç akışı dört fazdan oluşur (Yapılandırma, Çıkarım, Öbekleme, Değerlendirme).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

B_j	A block
C	PLS matrix
C_d	Multiway mixture assigments
D	Number of dimensions
d	Pitman-Yor process discount parameter
Dir	Dirichlet distribution
E	A sample set of partitionings or feature allocations
F	A feature allocation
$f(Z)$	Statistic for a partitioning Z
$\mathcal{G}$	Gamma distribution
$H(PROJ(Z,S))$	Projection entropy of a partitioning Z onto a subset S
$H(Z)$	Entropy of a partitioning Z
i	Sample index in the dataset
k	Mixture component index
K	Number of mixture components
K^+	Number of non-empty mixture components
L	Number of PLS parameters
$Mult$	Multinomial distribution
$[n]$	The set of integers 1 up to n
N	Number of samples in the dataset
$\mathcal{N}$	Gaussian distribution
$p(\cdot)$	Probability distribution
$\mathrm{pei}_n(B)$	Per-element information of a block B
$\mathcal{PO}$	Poisson distribution
$PROJ(Z,S)$	Projection of a partitioning Z onto a subset S
S	A subset of elements
s	Segment size in a block
V	Variance

X	Data tensor
x_i	Observed samples
z_i	Mixture assignments
Z	A partition
α	Dirichlet distribution concentration parameter
Δ	Number of additional dimensions
$\Delta_{i,k}$	A COD matrix
ϵ	Gaussian error
θ_k	Mixture component parameters
Θ	Latent tensor
ι	Sample index in the data tensor
κ	Parameter index in latent tensor
λ_k	Mixture component precision
π	Mixture proportions
$\pi(Z)$	Probability distribution of a partitioning Z
σ	A permutation of elements
Φ	A population of subsets of samples)
$\Phi_k(Z)$	Cumulative statistic of a partitioning Z
Ψ	A population of subsets of elements)
$\mid$	... conditioned on ...
$\sim$	... is distributed according to ...
∞	Infinity
$\vdash$	... is a partitioning of ...

LIST OF ACRONYMS/ABBREVIATIONS

CLUS	Clustering phase of CnG
CnG	CLUSTERnGO
COD	Cumulative occurence distribution
CONF	Configuration phase of CnG
CRP	Chinese restaurant process
DAG	Directed acyclic graph
DP	Dirichlet process
DPM	Dirichlet process mixture
EA	Entropy agglomeration
EVAL	Evaluation phase of CnG
FDR	False discovery rate
GO	Gene ontology
IBP	Indian buffet process
IMM	Infinite Multiway Mixture
IMPLS	Infinite Mixture of Piecewise Linear Sequences
INF	Inference phase of CnG
MAP	Maximum a posteriori
MCMC	Markov chain Monte Carlo
MLE	Maximum likelihood estimation
pdf	Probability density function
PDP	Poisson-Dirichlet process
PE	Projection entropy
PLS	Piecewise linear sequence
PYP	Pitman-Yor process
SBP	Stick breaking process
TS	Temporal segmentation in CONF
TSC	Two-stage clustering in CLUS

1. INTRODUCTION

In this thesis we focus on the generic problem of extraction of reliable structural information from noisy datasets, in particular time series data obtained from large scale gene expression experiments. The time series are observations of the expression levels of over 5600 genes at irregular times obtained from a yeast cell using microarray technology.

The greater scientific question in all these gene expression studies is the detection of specific interactions that occur among various groups of genes. Identification of these interactions helps in revealing the structure of the metabolic, regulatory or signalling mechanisms that are vital parts of a cell's life cycle. These highly complex biological mechanisms are often represented as networks; however even at the current relatively advanced technological level, precise information about these mechanisms can be obtained only via indirect measurements, hence statistical modelling techniques must be heavily employed for drawing valid conclusions from experimental data. Conclusions must be validated in the face of already established biological facts; this also requires that inferences must be summarized and described in a concise manner.

In this context, one particularly popular and useful approach to summarize the vast amount of experimental data is based on clustering. Here, the key idea is that the expression levels of related genes will exhibit a similar behavior and this similarity provides some important clues about the functioning of the underlying biological system. However, for drawing scientifically relevant and reliable conclusions, both the concept of similarity and model validation needs to be based on rigorous and generic methodology.

In contrast to other data analytics domains where machine learning methods are applied, biology and bioinformatics as scientific disciplines have much more stringent requirements in terms of validation of results from data analysis. Therefore, we have based our statistical methodology on Bayesian nonparametrics. Bayesian nonpara-

metrics provides a computational framework that can answer several questions such as model selection, data fusion, characterization of uncertainty, and tailored model development to reflect prior knowledge about the experimental conditions.

Based on this approach to clustering, we have developed and tested a full processing pipeline and the associated statistical methodology for time series data. The pipeline implemented in our application CLUSTERnGO (CnG) consists of four phases that also provide the framework of the questions we have studied.

The first two phases of our approach are the *Configuration* and *Inference* phases. Here, configuration refers to the selection of a particular model and inference refers to the computation of possible partitionings and other nuisance parameters. In the thesis, we have developed two novel Bayesian nonparametric models: Infinite Multiway Mixture (IMM) that extends the standard infinite mixture model; and Infinite Mixture of Piecewise Linear Sequences (IMPLS) that assumes a specific structure for its mixture components, tailored towards gene expression time series. In the inference phase, a Markov Chain Monte Carlo (MCMC) algorithm takes a gene expression data set and computes a sample set of likely partitionings of the elements (genes or proteins, etc.) conditioned to the given dataset. Technically, this is achieved by sampling from the posterior distribution over possible partitionings.

The third phase is the actual *Clustering* phase. Here, a clustering algorithm is employed to process the obtained sample set of partitionings in order to find particular clusters of elements (genes or proteins, etc.); all clusters that are sufficiently likely according to certain thresholds for similarity and dissimilarity. In CnG, Two-Stage Clustering is implemented as a heuristic solution for this phase. To find a better solution, we took a theoretical approach to this question and developed *cumulative statistics* and *entropy agglomeration* (EA) in order to express and summarize sample sets of partitionings and feature allocations. In order to evaluate the EA method, we also designed separate experiments outside of the biological domain, where the paragraphs of a literary text (a famous novel from 1922: Ulysses by James Joyce) were put to analysis to reveal the contextual relations among its words.

The final phase is the *Evaluation* phase. Here, the relevance of resulting clusters are evaluated by comparing them against previous biological knowledge. CnG implements standard multiple hypothesis testing of clusters with respect to a Gene Ontology database. In this way, we were able to detect clusters that can significantly represent particular Gene Ontology terms. The entire processing pipeline is implemented as a software package and released under GNU General Public License. The software has already been used by other researchers and cited shortly after its release.

1.1. What is clustering?

Clustering algorithms are arguably among the most important and popular tools of machine learning. However, clustering as a mathematical problem is by no means an elementary one. In fact, it is an ill-posed problem that presents many difficulties. The problem emerges by asking the following question: How can we group a given set of elements —based on what we know about them— into a set of clusters (where "cluster" refers to a subset of similar elements)? The computational answer to this question is to design a clustering algorithm, which will be able to perform the desired operation automatically in a satisfactory manner. A clustering algorithm, implemented as a software program, takes the information supplied about a set of observed elements as its input, and produces a set of resulting clusters to which these elements are assigned as its output.

Clustering algorithms are employed in many application fields as they constitute one of the basic means for extracting and summarizing information from datasets. Moreover, as indicated by the rise of big data research in machine learning, there is an emerging need to develop generic methodologies for rendering vast amounts of available data practically accessible and interpretable for various uses. Thus, it is an important task to formulate the clustering problem in a consistent and general manner. Such a formulation would contribute to theory —which can thus achieve better comprehensions of various other problems by comparing them with the clustering problem— as well as practice —which can thus better summarize data and render it more accessible for its purposes.

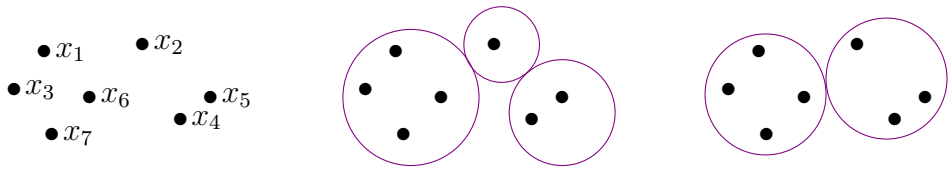
Clusters may represent different things depending on the context of the application field. For example, in natural language processing, they may represent topics that consist of word-elements; in bioinformatics, metabolic processes that consist of gene-elements; in sociology, social categories that consist of person-elements; in epidemiology, health conditions that consist of symptom-elements, and so on. Since they are solutions to a generic problem, clustering algorithms are applicable in any knowledge discipline by treating its building blocks as the data elements to be clustered.

In this thesis, we address specific questions about the clustering problem and present a clustering application CLUSTERnGO (CnG) for gene expression analysis. The next section provides a general description of the clustering problem, and the sections that follow introduce questions addressed in the rest of the thesis.

1.2. What is a clustering algorithm?

A *cluster* designates a subset of elements that are similar to one another. A *clustering algorithm* refers to an algorithm that can obtain subsets of similar elements from a given dataset. We call these subsets the "resulting clusters" found by the algorithm. Clustering algorithms may use any means to obtain the resulting clusters, but it is very common that they consist of several intermediate procedures, and one should generally expect that these intermediate procedures are built on a few basic atomic operations: *insert element into cluster, remove element from cluster, compute quantity for cluster, compute quantity for element, compute quantity for the clustering configuration, etc.*

As a practical guideline, a clustering algorithm is considered to be successful if each of the resulting clusters is similar within itself and different from the others. In effect, algorithms that can obtain successful results more frequently are considered to be better than the other algorithms. However, such guidelines remain vague, and one cannot validate and compare the criteria that justify the use of a clustering algorithm, unless these guidelines are supported by theoretical definitions of one's objectives.



(a) Observations **(b) Clusters A** **(c) Clusters B**

Figure 1.1. Algorithmic clustering assigns the observed samples to clusters.

Let us begin with an illustrative example. See Figure 1.1a, which shows a simple dataset to be clustered. There are seven observed samples $x_1, \ldots, x_7$. An *observed sample* designates an instance of any kind of mathematical object, provided that these instances constitute a sample set in a well-defined sample space, which is in this case the real plane $\mathbb{R}^2$. The clustering algorithm is expected to group the observed samples according to their similarities. For samples from $\mathbb{R}^2$, it is appropriate to use the Euclidean distances between sample pairs as a general measure of dissimilarity between samples. On the real plane, one can simply assume that the smaller the distance between the elements, the closer they are to one another, and therefore the more similar they are. For example we can see on Figure 1.1a that the samples x_4 and x_5 are close to each other. Given the sample space, this implies a smaller Euclidean distance, and it can be interpreted as a reliable indication of their similarity. In comparison we can say that the samples x_3 and x_5 are not close, there is a greater distance between them, and they are dissimilar to one another, as implied by a parallel reasoning.

It is expected that the clustering algorithm will *assign* the given samples to a number of clusters. Assume that our algorithm resulted in the clusters shown in Figure 1.1b, where seven samples are assigned to three clusters $\{x_1, x_3, x_6, x_7\}, \{x_2\}, \{x_4, x_5\}$. Even if we haven't defined an objective function that quantifies this result we can still check whether it accurately represents the similarity relations among these samples by looking at the distances between samples. For instance, samples in the cluster $\{x_4, x_5\}$ are similar to one another, so it looks like an accurate result to get from a clustering algorithm. For example if it was $\{x_3, x_5\}$ instead, since these two samples are not similar to one another, it would be an inaccurate result to get from a clustering algorithm. But depending on the sample space used and the notion of similarity employed in the approach, such simple criteria may not be available. A variety of metrics and functions

have been developed in order to measure the efficiency of clustering results for different problems, but there is no widely accepted standard, so we have to consider *clustering* as an abstract notion to be concretized according to the application question at hand.

The output of a clustering algorithm is usually assumed to cover all elements, i.e. no element can remain unassigned. It is also generally assumed that each element will be assigned to only a single cluster, i.e. clusters must not overlap. Mathematically, this means that the clustering assignments determine a *partitioning*. If these assumptions are removed the clustering assignments will determine a *feature allocation*, which is a superclass of partitionings. For example, $\{x_1, x_3, x_6\}, \{x_2, x_5\}, \{x_4, x_5\}$ is a feature allocation that is not a partitioning, because x_7 remains unassigned, and two clusters overlap at x_5. In practice it is more convenient to keep these two assumptions and work on partitionings since they involve fewer number of possibilities. It would be much more complicated to take into account feature allocations in general.

Most clustering algorithms are *iterative*, i.e. given the observations, the algorithm will run over several consecutive iterations until it produces the final output. Usually in such a scheme, each iteration receives the clustering assignments from the previous iteration, and runs a procedure that is expected to modify these assignments in a way to improve them, for instance from Clusters A to Clusters B in Figure 1.1. In other words, each iteration *reassigns* the elements into a new clustering configuration that may or may not be better than the previous one. After several iterations that seek to optimize the clustering assignments, the final assignment will be displayed to the user as the output of the clustering algorithm.

Almost every clustering algorithm can be expressed as an iterative procedure that optimizes a certain *objective function*. To justify the objective function used in the clustering algorithm, researchers naturally need conceptual frameworks to articulate and clarify their research questions. However, once the objective function is algebraically defined, and once an iterative procedure that efficiently optimizes this objective function is algorithmically designed, it becomes possible to formulate the principles that effectively justify the clustering algorithm in a way that is perfectly indifferent to the

conceptual approaches that may have initially informed the research question. That's why one must always distinguish the application point of view to a clustering problem from the theoretical point of view to the clustering problem.

From the application point of view, there are as many clustering problems as there are datasets to be clustered. Each dataset represents its own experimental research context that is going to inform any clustering approaches that its own requirements may demand. From the theoretical point of view, however, one can only speak of a clustering problem by defining an algebraic objective and designing an algorithmic solution, and once this is done, there is only a single clustering problem in question and nothing else. The two main purposes of this reduction are (1) to obtain a typical formulation that simplifies the statement of the problem, and (2) to open up a broad space of possible solutions that has already been developed and tested in the machine learning literature. Before getting to the thorough description of our Bayesian theoretical approach, we would like to illustrate the problem in the context of a popular algorithm: K-means.

Let $x_1, \ldots, x_N$ be a set of observed samples in a state space. And let the number of clusters K be pre-determined. K-means algorithm minimizes this objective function:

$$F(x_1, \ldots, x_N) = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - c_k\|^2 \tag{1.1}$$

where C_k denotes the kth cluster, $\|\cdot\|^2$ designates a norm in the state space, and

$$c_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i \tag{1.2}$$

is the kth cluster centroid. At the beginning of K-means clustering, the K cluster centroids have to be initialized at K points in the state space of observed samples. This initialization of cluster centroids can be random or algorithmically determined. In the iterative procedure of K-means clustering, the following two steps (illustrated in Figure 1.2) are repeated until the cluster centroids converge to unchanging positions:

Reassign the samples **Move the centroids**

Figure 1.2. K-means clustering is an iterative algorithm that consists of two steps.

1) Reassign each sample to the closest centroid (according to the given norm).
2) Move each cluster centroid to the mean value of its assigned samples.

K-means clustering provides a fairly simple description and a quite efficient solution for the clustering problem. But it also displays several shortcomings. Firstly, K is a very central high-level parameter in the clustering problem that should not be left to supplementary procedures that will run outside of the main clustering algorithm: It must have a logical formulation that takes K into account. Secondly, the initial positions of the K centroids have a huge effect on where they will convergence at the end, and therefore these initial positions should not be left to supplementary procedures outside the main algorithm either: It must provide a flexibility that can cover the whole state space. Thirdly, even if a cluster is defined to be "a subset of elements that are similar to one another," this does not automatically imply that a cluster can be represented by a centroid to which samples will be assigned depending on their distances to this centroid according to a norm in the state space. The notion of similarity can be defined in several ways, each of which implying a different representation for the clusters. For these reasons, we are going to investigate the clustering problem theoretically through Bayesian nonparametrics. The next chapter provides a technical introduction on infinite mixture models. The following sections give brief introductions for the contributions of the thesis.

1.3. Multiway clustering

When the sample set is modeled by an infinite mixture model, the independence among the sequence of observations $x_1, \ldots, x_7$ is represented by the infinite random atomic sequence of mixture proportions and parameters drawn from the nonparametric

Dirichlet prior: $(\pi_1, \theta_1), (\pi_2, \theta_2), (\pi_3, \theta_3), \ldots$. But in some cases, the observed samples might be indexed in more than one dimension. For instance, we might be modeling a matrix $\{x_{1:N_1,1:N_2}\}$ or a tensor $\{x_{1:N_1,1:N_2,1:N_3}\}$ of observations. In such cases, the assumption of independence among the observations does not naturally follow from the indexing scheme of the observations. One can naturally assume that the sequence of indices in each single dimension (either $1, \ldots, N_1$ or $1, \ldots, N_2$) should represent an independence among its particular index values $1, 2, 3, \ldots$. However, among the separate dimensions represented by the indices i, j, k of $x_{i,j,k}$, there remains a dependence relation embedded in the tensor of observations: how to handle the multiway dependence among the dimensions of a tensor?

In the case of an observation vector $\{x_i\}$ with a single dimension, we can take its index i to distinguish among the independent observations. In some datasets, an observation tensor X with more than one index may be necessary. We address this question by devising a multiway representation: Infinite Multiway Mixture (IMM), where an observation tensor X is modeled by a smaller latent tensor Θ, by putting nonparametric priors on each of its dimensions. In addition, Θ is taken to be a factorization of M latent tensors Θ_m, in order to model the dependence among these dimensions.

1.4. Time-series modeling

A practical clustering problem is how to cluster a set of time-series samples. In this problem, each observed sample is an independent M-dimensional vector that represents values that were observed at certain time points: $x_{1:N,1:M}$ where M designates the length of a single observation. In certain gene expression analysis problems, microarray experiments are designed so that the expression levels of the N genes of interest are observed at M particular time points thoughout the experiment process. These time points may be unequally distributed during the experiment, so that subsequent time points may be seconds, minutes, or even hours apart from each other. In order to model a sample set of such time-series data by an infinite mixture model, we present a method that introduces a particular covariance structure for the mixture component likelihoods.

This modification applies a matrix multiplication that structures each mixture component mean vector as a piecewise linear sequence (PLS), which involves predetermined segments of subsequent time points, each segment maintaining a constant slope. By applying this matrix transformation, we effectively state that the sequence of expression values for a single gene $x_{i,1}, \ldots, x_{i,M}$ are *not* independent. Since we state that these values maintain constant slopes along certain segments of time points, we do not leave all of their covariance to the implicit Gaussian noise in the mixture model, but we represent some of their covariance by the transformation matrix. We call this model, an infinite mixture of piecewise linear sequences (IMPLS). This model is also employed in the bioinformatics application CLUSTERnGO.

1.5. Summarizing a combinatorial sample set

Dirichlet process was initially defined as the stochastic process all of whose measurable partitions are Dirichlet-distributed [1]. Although this definition remains the basic theoretical reference and the source of its name, various alternative formulations of this stochastic process have been developed since then [2]. The invention of Chinese restaurant process, one of the best-known iterative constructions of the Dirichlet process, historically depended on a formula from the population genetics literature: Ewens [3] (with the elaboration of Watterson [4]) had devised a method for sampling distributions of selectively neutral alleles. By taking the probability distribution formula from Ewens' method and carrying out a mathematical and logical analysis, Kingman [5] described the meaning of this method by defining it as an *Ewens structure*, which he classified as a *partition structure*, a concept he constructed by relying on the theory of partitions [6]. Chinese restaurant process (CRP), relies theoretically on this concept [7].

However, the concept of partition structure has certain limitations. Kingman [5] defined 'representable' partition structures as those which can be written as the integral of a continuous function with respect to a probability measure on the infinite dimensional simplex. Ewens structure is a representable partition structure, but not all partition structures are representable in this sense. An instance of a 'non-representable'

partition, pointed out to him by Watterson, is the distribution that concentrates on the partition in which all allele types only appear once. In this thesis, we provide an alternative conceptual formulation for partitionings that also applies to feature allocations. In clustering, the clusters are usually assumed to constitute a *partitioning* of data elements, i.e. a set of disjoint subsets —called 'blocks'— that cover each of the given elements once. In other cases, where clusters are allowed to repeat or omit some of the elements, they constitute a *feature allocation*, a superclass of partitionings. We devise the basics for a combinatorial statistics methodology that we call cumulative statistics. A practical algorithm that we propose based on this concept is entropy agglomeration (EA).

The contributions of the thesis are distributed to the chapters as follows:

- Infinite Multiway Mixture with Factorized Latent Parameters (Chapter 3)
- Infinite Mixture of Piecewise Linear Sequences (Chapter 4)
- Cumulative Statistics and Entropy Agglomeration (Chapter 5)
- Clustering Words by Entropy Agglomeration (Chapter 6)
- CLUSTERnGO: Cluster analysis of gene expression profiles (Chapter 7)

Chapter 8 concludes the thesis.

2. BAYESIAN CLUSTERING AND INFINITE MIXTURE MODELS

Bayesian theory provides a principled approach to do model-based inferences for clustering and other machine learning purposes. There is a large theoretical literature on Bayesian inference methods with several models and approaches to all kinds of machine learning problems. But we restrict our focus to the use of Bayesian mixture models in the clustering problem. This focus allows us to elaborate the development of our contributions more strictly in terms of both theory and practice. The fact that there are so many texts in the theoretical literature of Bayesian methods owes partly to the practical usability of these methods and partly to the simple logic that they rely on, namely, the Bayes rule. This is why we would like to begin our presentation with a description of this rule. It is stated as follows: *Posterior is proportional to prior times likelihood.* In the statement, the three terms *posterior*, *prior* and *likelihood* refers to probabilities, i.e. quantities that take values between zero and one, quantities that are functions of certain probabilistic events. The other terms *proportional* and *times* in the statement refers to simple algebraic multiplicative operations. The mathematical formulation of the Bayes rule can be written as follows:

$$P(A \mid B) \;\propto\; P(A)\; P(B \mid A) \tag{2.1}$$

There are two events that follow one another: First A happens. Then B happens. In the formula '$P(A \mid B)$' represents the posterior, '$P(A)$' the prior and '$P(B \mid A)$' the likelihood. The proportionality of the posterior is expressed by the symbol '$\propto$' and the quantity "prior times likelihood" is expressed simply by writing these functions side by side: "$P(A)\; P(B \mid A)$." One must indicate that this logical formula determines all the possible meanings and interpretations of the three quantities that it relates. Even when the quantities are independently determined each by a formula of its own, their logical relation and proper interpretation are ultimately decided by the Bayes rule.

In this way, a temporal relation between two events A and B can be formulated in order to compute their probabilities. One can also express the Bayesian probabilistic relation of several events, if it is possible to express their relation in terms of the Bayes rule. Integrations over probabilistic functions and various analytical manipulation of these integration formulas allow us to express many complex relationships in terms of the Bayes rule. Hence the vast theoretical literature of Bayesian statistics.

These events related by the Bayes rule can be organized in a generative model. In a generative model, each event represents the initialization of a variable. In our example: In the first step, A is set to a value, call it a. In the second step, B is set to a value, call it b. Their dependence is given by the Bayes rule. In this way, a variable A is symbolically differentiated from the value a that it is instantiated with. This differentiation in each variable is reflected to the three terms of the formula, so that $P(A = a \mid B = b)$ represents a particular probability from the posterior, $P(A = a)$ represents a particular probability from the prior and $P(B = b \mid A = a)$ a particular probability from the likelihood. These particular probabilities must not be confused with the general probabilities that take part in the Bayes rule. To illustrate this for a single variable: Depending on the mathematical expression of the variable A, its prior $P(A)$ can be a vector, it can be a matrix, it can be a tensor. But $P(A = a)$ expresses a single quantity between zero and one that represents our belief on the probability of A being instantiated with a.

The general probabilistic relation expressed by the *Bayes rule* can be represented in a more exact algebraic equation that we call the *Bayes theorem* or the *Bayes law*:

$$P(A \mid B) \;=\; \frac{P(A)\; P(B \mid A)}{P(B)} \tag{2.2}$$

In this version of the Bayes rule the factor of proportionality $P(B)$ is explicitly represented as the denominator on the right hand side. It is especially crucial to logically distinguish $P(A)$ and $P(B)$ despite the apparent symmetry implied by the equation. Both can be called 'marginals' because both involve a single variable. But the main difference is that the prior $P(A)$ is given by the model, whereas the factor of propor-

tionality $P(B)$ at the denominator must be computed as a summation over a:

$$P(B) \;=\; \sum_{a} \; P(A=a)\; P(B \mid A=a) \tag{2.3}$$

This is the most general expression of the Bayes theorem that underlies all Bayesian methodology. To illustrate this equation, let us define a generative model with two variables: Let $\theta \in \mathbb{R}^2$ be an underlying parameter that we cannot observe. Let $x \in \mathbb{R}^2$ be an observed sample that depends on the parameter θ. Let $p(\theta)$ be the prior for the hidden parameter θ and let $p(x|\theta)$ be the likelihood which represents our belief on how the observed sample x depends on the hidden parameter θ. So we have:

$$p(\theta|x) \;=\; \frac{p(\theta)\; p(x|\theta)}{p(x)} \tag{2.4}$$

In this equation, the lowercase $p(\cdot)$ designates a probability density function over $\mathbb{R}^2$. Given the prior $p(\theta)$ and the likelihood $p(x|\theta)$, we can use the Bayes rule to infer the corresponding posterior $p(\theta|x)$. The generative model consists of two steps:

$$\begin{aligned} \theta \;&\sim\; p(\theta) \\ x \;&\sim\; p(x|\theta) \end{aligned} \tag{2.5}$$

The only evident fact is that we have observed a sample x_1 as the value for the variable x. By modeling the variable x, we assume that the sample x_1 was generated according to the model written above. To write explicitly: In the first step, a parameter θ_1 is drawn from the prior $p(\theta)$. In the second step, a sample x_1 is drawn from the likelihood $p(x|\theta=\theta_1)$. The prior $p(\theta)$ for the parameter θ and the likelihood $p(x|\theta)$ that relates the parameter variable θ to the observation variable x is illustrated on Figure 2.1 in grey dashed ellipses. Based on the fact that x_1 is observed and the model assumptions, we can infer the posterior $p(\theta|x=x_1)$ for the parameter variable θ given our observation $x=x_1$. The posterior $p(\theta|x=x_1)$ and the marginal $p(x)$ are illustrated on Figure 2.1 in blue dashed ellipses. As in the previous formulation, the denominator $p(x)$,

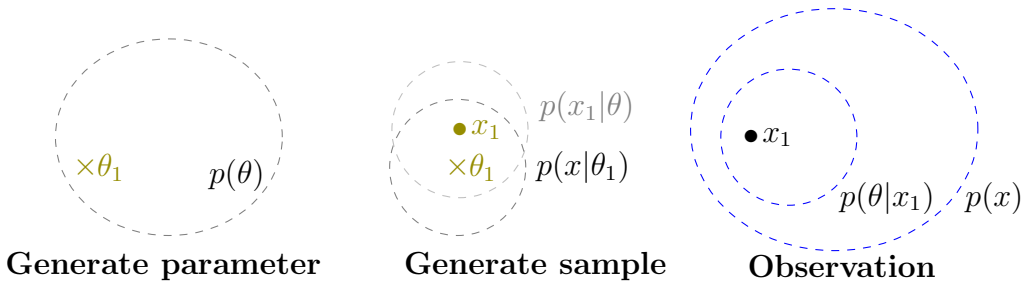


Figure 2.1. Simple generative model with two variables.

sometimes called *evidence* or *normalization constant*, is computed by an integration:

$$p(x) \;=\; \int p(\theta)\; p(x|\theta)\; d\theta \tag{2.6}$$

If we extend this model to incorporate N observations that depend on K parameters, we can have a mixture model that involves K mixture components. Our initial example with 7 observations drawn from 3 parameters is shown in Figure 2.2. To express this generation of variables as a mixture model, another set of variables called sample assignments $z_1, \ldots, z_7$ will be needed. Let us just state that it is well known that the K-means clustering algorithm as we described above relies on such a mixture model as its generative model (For a detailed explanation, see [8]). If there is a predetermined number K of mixture components in the model as in K-means clustering, it's called a *finite mixture model*. To evade this constraint, *infinite mixture models* are used. An infinite mixture model has an infinite number of mixture components in theory, but when actually generating samples from an infinite mixture model, almost all of these mixture components will be empty, since only a finite number of samples can be generated in finite time.

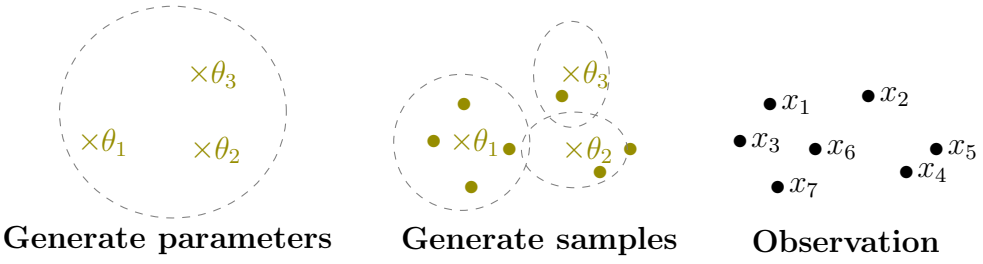


Figure 2.2. A model that generates 3 parameters and 7 observations from them.

An infinite mixture model is conventionally derived by taking the limit of a finite mixture model as the number of mixture components goes to infinity [9]. This generates an infinite sequence of parameters and an infinite sequence of mixture proportions. In the conventional representation, these two sequences are matched in an infinite sequence of atomic samples that make up a random measure, called a *mixing distribution*. *Dirichlet process* (DP) is the best-known Bayesian prior that generates such a mixing distribution. It's called a *nonparametric prior* due to the infinite sequence of parameters it generates. Iterative schemes for DP like Polya urn process [10] and Chinese restaurant process [7] are useful in devising Bayesian inference methods for such infinite atomic sequences over a finite quantity of observed samples. In the next section, we explain finite mixture models, derive infinite mixture models, and recite basic Monte Carlo inference methods based on the iterative constructions of the infinite mixture models.

2.1. Mixture models

Mixture models are among the most common probabilistic models used for clustering. Basically, they are generative models. A generative model, sometimes called a Bayesian network, is often defined as an interdependent DAG (directed acyclic graph) of random variables. On this graph, every node denotes a random variable, and every edge denotes the dependence of a random variable to another random variable. Each random variable on the graph is generated either by a likelihood or by a prior pdf (probability density function). Variables that depend on other variables are generated by likelihoods, whereas variables that do not depend on any other variable are generated by priors. For example, in a mixture model, π and $\tilde{\theta}$ do not depend on other variables, so they are generated by priors, whereas z_i and x_i are generated by likelihoods, because they depend on other variables, as shown in Figure 2.3.

A special convention in these graphs is the plate notation. A plate is a rectangle around a group of nodes. It indicates that each of these nodes represents several random variables. A node in a plate is written with an index, and a range of numbers is shown on the plate. This indicates that the node's index takes values in that range, so that

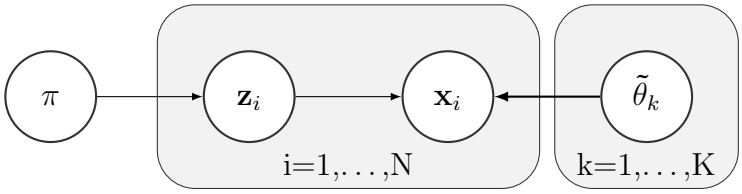


Figure 2.3. Graphical notation for a mixture model

the model contains distinct variables for each value in this index range. For example, in Figure 2.3, the node x_i represents N variables $x_1, x_2, \ldots, x_N$. When an edge is between nodes on different plates, it's assumed that all of the variables represented by both sides are connected, as the case with x_i and $\tilde{\theta}_k$ on the graph. The graph shows the dependencies between variables, but the generative model can fully be represented only when the likelihoods and priors that generate the random variables are defined. These are written for the mixture model as follows:

$$\begin{aligned} \pi &\sim Dir(\pi|\frac{\alpha}{K},\ldots,\frac{\alpha}{K}) \\ \tilde{\theta}_k &\sim H(\tilde{\theta}_k) \\ z_i \mid \pi &\sim Mult(z_i|\pi) \\ x_i \mid \tilde{\theta}, z_i &\sim \prod_{k=1}^{K} F(x_i|\tilde{\theta}_k)^{z_{ik}} \end{aligned} \tag{2.7}$$

On each line of this definition, a variable and its dependencies are shown on the left, and a pdf is shown on the right hand side of the symbol '$\sim$', which means that the variable is distributed according to that pdf, i.e. the variable is drawn from that pdf. In the mixture model defined above, Dir and $Mult$ are known pdfs, whereas H and F are model-specific pdfs that need to be defined. The product of all the likelihoods and priors in a generative model gives the joint pdf of all variables. This product written as an equation is called a factorization of this joint pdf. Other pdfs can be derived from this factorization using integration and Bayes rule. Before making inference on the generative model, it needs to be decided which variables are to be inferred, then a pdf from the model's factorization is derived and chosen to be the target pdf.

There are different kinds of Bayesian inference methods. If the pdf of interest can be computed numerically, it's called exact inference. In exact inference, the terms of a factorization are actually multiplied and summed up to compute the results of the integrals. Exact computation is usually not feasible, so approximate inference is used in most cases. There are inference methods to obtain optimal samples from a pdf. Variational inference methods are optimization algorithms that find samples at the peaks of a target pdf. They are called maximum likelihood estimation (MLE) or maximum a posteriori (MAP) depending on the chosen target pdf.

In this thesis, Markov chain Monte Carlo (MCMC) methods are used for Bayesian inference. These methods are not optimization algorithms that settle on a single sample after some iterations. Instead, they are sampling methods that generate a set of samples as a representative group for the pdf of interest. One obtains a different sample set for each time an MCMC method is run, but all of these sets are approximately distributed according to the chosen target pdf. If an MCMC method is run for a short time, one can still get a sample set that roughly represents the target pdf. If it's run for a longer time, one obtains a sample set that more precisely represents the target pdf. Gibbs sampling, a common method, designates a specific class of Metropolis-Hastings algorithms where each variable is sampled from its full conditional given all the other variables. In a single iteration of Gibbs sampling, each of the variables are sampled one by one, according to its own full conditional. After iterating MCMC many times, the samples obtained at these iterations can be used as a sample set to approximately represent the target pdf. To apply Gibbs sampling on a generative model, full conditionals need to be derived for all variables that will be inferred. If this is not possible, one can still design a Metropolis-Hastings method that does not require this derivation. For more information about generative models and inference methods, please refer to the book "Pattern Recognition and Machine Learning" by Christopher M. Bishop [8]. Dirichlet process mixture (DPM) is one of the most fundamental models in Bayesian nonparametrics: it's a mixture model with infinite number of mixture components. It can be used for clustering by interpreting its mixture components as clusters. The prior of a DPM is a Dirichlet process (DP). In the following part, a finite mixture model with K mixture components is defined, and how to generate samples from it. Then, the full

conditionals are derived to be used in a Gibbs sampler, and finally, mixture parameters are integrated out to derive full conditionals to be used in a collapsed Gibbs sampler. In the next part, we define the infinite mixture, which is obtained by bringing K in the finite mixture to infinity. This new mixture with infinite number of mixture components is a DPM. Then, three different formulations of DPM are examined. We first explain the Polya Urn Scheme, the associated sample generation method and full conditional to use in Gibbs sampling. Then, we explain the Chinese Restaurant Process (CRP), which has a similar sample generation method. For CRP, full conditionals are derived for both Gibbs sampling and collapsed Gibbs sampling. Finally, Stick Breaking Process and a possible sample generation method is described. Although there are many inference methods for DPM that have been proposed in recent years, here we discuss the most basic MCMC methods that were summarized by R. Neal in 2000 [11]. The conventions for the variables of DPM are taken from an encyclopedia article of Dirichlet process by Y. W. Teh [12]. In the following two subsections we explain basic generation and inference procedures for finite mixture models and infinite mixture models in detail.

2.1.1. The finite mixture model

Imagine a simple model, where an observed variable x only depends on a hidden parameter $\tilde{\theta}$ through a likelihood function $F(x|\tilde{\theta})$. In this simple model, $\tilde{\theta}$ would be inferred directly by using the evidence x, the prior of $\tilde{\theta}$ and the given likelihood function. A mixture model is similar to this simple model, except that there are several values for the hidden parameter, denoted as $\tilde{\theta}_k$ where $k = 1, \ldots, K$. Each of these K parameters $\tilde{\theta}_k$ define a component in the mixture. In a mixture model, in addition to the observation x, another variable is needed: the sample assignment z that determines which parameter generates x by assigning x to one of the K mixture components.

2.1.1.1 The likelihoods and the mixture density. Let's say there are N observed samples x_i where $i = 1, \ldots, N$. If sample x_i belongs to the kth component of the mixture, it is assumed to be distributed according to the corresponding likelihood

function:

$$p(x_i|\tilde{\theta}, z_{ik} = 1) \;=\; F(x_i|\tilde{\theta}_k) \tag{2.8}$$

In this equation, the sample assignment z_i is a binary vector with K elements. When $z_{ik} = 1$, all other elements z_{ij} are zero, and this means that the sample x_i is assigned to the kth component of the mixture.

$$z_{ik} = \begin{cases} 1 & \text{if observation } x_i \text{ belongs to mixture component } k \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

Thus, before drawing the samples x_i, one must draw the assignments z_i to which they depend on. Before drawing z_i, one must determine π_k as the prior proportions for each of the mixture components. These proportions must sum up to one:

$$\sum_{k=1}^{K} \pi_k = 1 \tag{2.10}$$

The sample assignments z_i are drawn from a multinomial distribution with the proportions vector $\pi = \{\pi_k\}$ as its parameter.

$$z_i \mid \pi \;\sim\; p(z_i|\pi) \;=\; Mult(z_i|\pi) \;=\; \prod_{k=1}^{K} \pi_k^{z_{ik}} \tag{2.11}$$

When the sample assignment z_i is known, x_i can be sampled from the mixture component it is assigned to:

$$x_i \mid \tilde{\theta}, z_i \;\sim\; p(x_i|\tilde{\theta}, z_i) \;=\; \prod_{k=1}^{K} F(x_i|\tilde{\theta}_k)^{z_{ik}} \tag{2.12}$$

The likelihoods over x_i and z_i are defined conditional to π and $\tilde{\theta}_k$.

$$z_i \mid \pi \ \sim p(z_i|\pi) \ = \ Mult(z_i|\pi) \ = \ \prod_{k=1}^{K} \pi_k^{z_{ik}}$$

$$x_i \mid \tilde{\theta}, z_i \ \sim p(x_i|\tilde{\theta}, z_i) \ = \ \prod_{k=1}^{K} F(x_i|\tilde{\theta}_k)^{z_{ik}} \tag{2.13}$$

If the likelihood of x_i is derived by integrating over its sample assignment z_i, one obtains the sample pdf $p(x_i|\tilde{\theta}, \pi)$ of a mixture model, which is also called a mixture density function:

$$p(x_i|\tilde{\theta}, \pi) = \sum_{z_i} p(z_i|\pi) \ p(x_i|\tilde{\theta}, z_i) = \sum_{k=1}^{K} \pi_k \ F(x_i|\tilde{\theta}_k) \tag{2.14}$$

The distributions of x_i and z_i only involve the likelihood probabilities, and are adequate to make a maximum-likelihood estimation (MLE). However, to find a maximum a posteriori (MAP) estimate, one needs the posterior distributions of x_i and z_i that depend on the priors of their parameters: priors over the mixture proportions vector π and the mixture component parameters $\tilde{\theta}_k$.

2.1.1.2. The priors and the complete generative model. The proportions vector π is drawn from a symmetrical Dirichlet distribution:

$$\pi \ \sim \ p(\pi) \ = \ Dir(\pi|\frac{\alpha}{K}, \ldots, \frac{\alpha}{K}) = \frac{\Gamma(\alpha)}{\Gamma(\alpha/K)^K} \prod_{k=1}^{K} \pi_k^{\frac{\alpha}{K}-1} \tag{2.15}$$

This distribution can have three different shapes:

(i) $\alpha = K$: A uniform, flat distribution.
(ii) $\alpha > K$: A unimodal dome, its mean at $\pi_k = \frac{1}{K}$.
(iii) $\alpha < K$: A round valley, K modes at its corners $\pi_k = 1$

For $\alpha > K$, α is called the pseudocount hyperparameter, because it behaves as if there were α number of previously observed samples that were distributed equally among the K mixture components. As the α parameter is further increased, the dome becomes taller, and the region around its mode $\pi_k = \frac{1}{K}$ becomes more and more pronounced. If K is relatively large, some of the K mixture components may be empty, i.e. they may not have samples assigned to them. Number of non-empty mixture components is denoted by K^+. For $\alpha < K$, α is called the diversity hyperparameter, because the greater the α value, the more non-empty mixture components exist, i.e. K^+ is greater. When α is increased, the valley becomes flatter, and its corners $\pi_k = 1$ become less pronounced, allowing a more diverse distribution of samples among the K mixture components. A mixture component parameter $\tilde{\theta}_k$ is drawn from H:

$$\tilde{\theta}_k \sim p(\tilde{\theta}_k) = H(\tilde{\theta}_k) \tag{2.16}$$

The prior $H(\tilde{\theta}_k)$ is usually chosen as the conjugate prior of the mixture component function $F(x_i|\tilde{\theta}_k)$. Finally, the complete generative model for the finite mixture is obtained by combining the previous equations:

$$\begin{aligned} \pi &\sim Dir(\pi|\frac{\alpha}{K},\ldots,\frac{\alpha}{K}) \\ \tilde{\theta}_k &\sim H(\tilde{\theta}_k) \\ z_i \mid \pi &\sim Mult(z_i|\pi) \\ x_i \mid \tilde{\theta}, z_i &\sim \prod_{k=1}^{K} F(x_i|\tilde{\theta}_k)^{z_{ik}} \end{aligned} \tag{2.17}$$

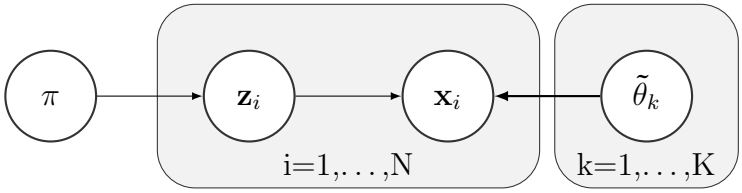


Figure 2.4. Bayesian graph for the finite mixture model

This model defines the generation process in two stages. Firstly, a particular mixture model is generated by drawing π and $\tilde{\theta}_k$ for $k = 1, \ldots, K$. Secondly, N observations from this model are generated by drawing z_i and x_i for all $i = 1, \ldots, N$. Dependencies among the random variables of this model can be clearly seen in Bayesian graph notation in Figure 2.4. Samples are generated by the following steps:

(i) Sample π from $Dir(\alpha)$
(ii) Sample $\tilde{\theta}_k$ from H for all $k = 1, \ldots, K$
(iii) Sample z_i from $Mult(\pi)$ for all $i = 1, \ldots, N$
(iv) Sample x_i from $\prod_k F(x_i|\theta_k)^{z_{ik}}$ for all $i = 1, \ldots, N$

2.1.1.3. Gibbs sampling for the finite mixture model. For applying Gibbs sampling on the mixture model, full conditionals are needed. To derive full conditionals, the factorization for the full joint pdf is written:

$$p(z, x, \tilde{\theta}, \pi) = p(\pi) \ \{\prod_{k=1}^{K} p(\tilde{\theta}_k)\} \ \{\prod_{i=1}^{N} p(z_i|\pi)\} \ \{\prod_{i=1}^{N} p(x_i|\tilde{\theta}, z_i)\} \tag{2.18}$$

By writing the likelihoods and priors, the expression for the full joint is obtained:

$$\begin{aligned} p(z, x, \tilde{\theta}, \pi) =& \{\frac{\Gamma(\alpha)}{\Gamma(\alpha/K)^K} \prod_{k=1}^{K} \pi_k^{\frac{\alpha}{K}-1}\} \ \{\prod_{k=1}^{K} H(\tilde{\theta}_k)\} \\ & \{\prod_{i=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{ik}}\} \{\prod_{i=1}^{N} \prod_{k=1}^{K} F(x_i|\tilde{\theta}_k)^{z_{ik}}\} \end{aligned} \tag{2.19}$$

In our example problem, the following functions will be used (P and Q are scalars):

$$\begin{aligned} H(\theta) &= \mathcal{N}(0, P) \\ F(x|\theta) &= \mathcal{N}(\theta, Q) \end{aligned} \tag{2.20}$$

The full joint becomes:

$$
\begin{aligned}
p(\tilde{\theta}, z, x, \pi) = \exp\{ &- \frac{D}{2Q} + \sum_{k=1}^{K} \{ \frac{S_k \tilde{\theta}_k}{Q} - \frac{1}{2}\, \tilde{\theta}_k^2\, (\frac{1}{P} + \frac{N_k}{Q}) \} \\
&+ \sum_{k=1}^{K} (N_k + \frac{\alpha}{K} - 1) \log \pi_k + \log \Gamma(\alpha) \\
&- K \log \Gamma(\frac{\alpha}{K}) - \frac{NK}{2} \log 2\pi Q - \frac{K}{2} \log 2\pi P \} \\
N_k = \sum_{i=1}^{N} z_{ik} \quad S_k = \sum_{i=1}^{N} z_{ik} x_i \quad D = \sum_{i=1}^{N} x_i^2 &
\end{aligned}
\tag{2.21}
$$

By choosing relevant terms from the joint pdf, the full conditionals for π, $\tilde{\theta}_k$ and z_i are obtained:

$$
\begin{aligned}
p(\pi \mid z) &= Dir(\pi \mid N_k + \frac{\alpha}{K}) \\
p(\tilde{\theta}_k \mid x, z) &= \mathcal{N}(\tilde{\theta}_k | m_k, V_k) \\
V_k &= \frac{Q}{N_k + Q/P} \qquad m_k = \frac{S_k}{N_k + Q/P} \\
p(z_i \mid x_i, \tilde{\theta}, \pi) &= Mult(z_i \mid \phi_i) \\
\phi_{ik} &= \frac{\pi_k}{\sqrt{2\pi Q}} exp(-\frac{(x_i - \tilde{\theta}_k)^2}{2Q})
\end{aligned}
\tag{2.22}
$$

Gibbs sampling algorithm for the mixture model is as follows:

(i) Sample π from $p(\pi \mid z)$

(ii) Sample $\tilde{\theta}_k$ from $p(\tilde{\theta}_k | x, z)$ for all $k = 1, \ldots, K$

(iii) Sample z_i from $p(z_i \mid x_i, \tilde{\theta}, \pi)$ for all $i = 1, \ldots, N$

(iv) Go to 1

2.1.1.4. Collapsed Gibbs sampling for the finite mixture model. For applying collapsed Gibbs sampling on the mixture model, $\tilde{\theta}$ is integrated out from the joint pdf to

obtain:

$$p(z,x,\pi) = exp(\frac{K-N}{2}\log\ 2\pi Q - \frac{K}{2}\log\ 2\pi P - \frac{D}{2Q} + \log\Gamma(\alpha) - K\log\Gamma(\frac{\alpha}{K}) + \sum_k\{(N_k+\frac{\alpha}{K}-1)\log\pi_k - \frac{1}{2}\log\ (N_k+Q/P) + \frac{S_k^2}{2Q(N_k+Q/P)}\} \tag{2.23}$$

By choosing relevant terms from the joint pdf $p(z,x,\pi)$, the full conditionals for π and z_i are obtained:

$$p(\pi \mid z,x) \ = \ Dir(\pi|N_k+\frac{\alpha}{K})$$

$$p(z_i \mid z_{-i},x,\pi) \propto \ exp\{\sum_k\{N_k\ \log\pi_k - \frac{1}{2}\log\ (N_k+\frac{Q}{P}) + \frac{S_k^2}{2Q(N_k+\frac{Q}{P})}\}\} \tag{2.24}$$

Collapsed Gibbs sampling algorithm for the mixture model is as follows:

(i) Sample π from $p(\pi \mid z)$
(ii) Sample z_i from $p(z_i \mid z_{-i},x,\pi)$ for all $i = 1,\ldots,N$
(iii) Go to 1

2.1.2. The infinite mixture model (DPM)

For obtaining the infinite mixture model, one needs an appropriate formulation to use when $K \to \infty$. Assume that π and $\tilde{\theta}_k$ are given. The sample likelihood is

$$p(x_i|\pi,\tilde{\theta}) = \sum_{k=1}^{K}\pi_k F(x_i|\tilde{\theta}_k) \tag{2.25}$$

Then, $G(\theta)$ is introduced as a summation of K weighted dirac delta functions in parameter space Θ:

$$G(\theta) = \sum_{k=1}^{K} \pi_k \delta(\theta - \tilde{\theta}_k) \tag{2.26}$$

and rewrite the sample density as

$$p(x_i|G) = \int F(x_i|\theta) G(\theta) d\theta \tag{2.27}$$

First formulation of a model:

$$p(x_i|\pi, \tilde{\theta}) = \sum_{k=1}^{K} \pi_k F(x_i|\tilde{\theta}_k) \tag{2.28}$$

Second formulation of the same model:

$$p(x_i|G) = \int F(x_i|\theta) G(\theta) d\theta \tag{2.29}$$

This new formulation is equivalent to

$$\begin{aligned} \theta_i \ &\sim \ G \\ x_i \mid \theta_i \ &\sim \ F(x_i|\theta_i) \end{aligned} \tag{2.30}$$

The new variable G now has infinite number of atoms

$$G(\theta) = \sum_{k=1}^{\infty} \pi_k \delta(\theta - \tilde{\theta}_k) \tag{2.31}$$

To have an infinite mixture model, a prior over G needs to be defined. It turns out that this prior is a Dirichlet process

$$G \ \sim \ DP(H, \alpha) \tag{2.32}$$

where H is the base distribution and α is the diversity parameter. The infinite mixture, or the Dirichlet process mixture:

$$\begin{aligned} G &\sim DP(H, \alpha) \\ \theta_i &\sim G \\ x_i \mid \theta_i &\sim F(x_i|\theta_i) \end{aligned} \tag{2.33}$$

It is equivalent to a mixture model where $K \to \infty$. It can be shown that the posterior of a DP behaves as follows:

$$\begin{aligned} p(G) &= DP(H, \alpha) \\ p(G|\theta_1) &= DP(\frac{\alpha}{\alpha+1}H + \frac{1}{\alpha+1}\delta(\theta_1)\ ,\ \alpha+1) \\ p(G|\theta_{1,2}) &= DP(\frac{\alpha}{\alpha+2}H + \frac{1}{\alpha+1}\sum_{j=1,2}\delta(\theta_j)\ ,\ \alpha+2) \\ &\dots \\ p(G|\theta_{1:i-1}) &= DP(\frac{\alpha}{\alpha+i-1}H + \frac{1}{\alpha+i-1}\sum_{j=1}^{i-1}\delta(\theta_j)\ ,\ \alpha+i-1) \end{aligned} \tag{2.34}$$

2.1.2.1. Generate samples using the Polya Urn Scheme. The definition of DP is used to find θ_i conditional to $\{\theta_{1:i-1}\}$

$$\theta_i \mid \theta_{1:i-1} \sim p(\theta_i|\theta_{1:i-1}) = \frac{1}{i-1+\alpha}\sum_{j=1}^{i-1}\delta(\theta_i - \theta_j) + \frac{\alpha}{i-1+\alpha}H \tag{2.35}$$

The value θ_i is either one of the previous values, or it's a new value:

$$\begin{aligned} \text{Pick one of } \theta_{-i} \text{ with probability } &\frac{1}{i-1+\alpha} \\ \text{Sample } \theta_i \text{ from } H \text{ with probability } &\frac{\alpha}{i-1+\alpha} \end{aligned} \tag{2.36}$$

Samples are generated by the following steps:

(i) Pick a new mixture component parameter θ_1 from H
(ii) Initialize $i = 2$
(iii) Pick an old θ_i from $\{\theta_{1:i-1}\}$ with probability $i - 1$, or
pick a new θ_i from H with probability α
(iv) Increment i, go to step 3

2.1.2.2. Gibbs sampling based on the Polya Urn Scheme. Full conditional for θ_i is obtained by combining with likelihood:

$$p(\theta_i \mid \theta_{-i}, x) \propto p(\theta_i \mid \theta_{-i})\ p(x_i \mid \theta_i)$$

$$p(\theta_i \mid \theta_{-i}, x) \propto \sum_{j \neq i} F(x_i|\theta_j)\ \delta(\theta_i - \theta_j) + \alpha\ P_i \int F(x_i|\theta)H(\theta)d\theta \tag{2.37}$$

Therefore,

$$\text{Pick } \theta_j \in \theta_{-i} \text{ with probability } \propto \mathcal{N}(x_i \mid \theta_j, Q)$$

$$\text{Pick } \theta_i \text{ from } P_i \text{ with probability } \propto \alpha\, \mathcal{N}(x_i \mid 0, Q + P) \tag{2.38}$$

where P_i is the posterior with single observation:

$$P_i = p(\theta|x_i) = \mathcal{N}(\theta|\frac{x_i}{1 + Q/P}, \frac{Q}{1 + Q/P}) \tag{2.39}$$

Gibbs sampling algorithm for Polya Urn Scheme is as follows:

(i) Sample $\tilde{\theta}_k$ from $p(\theta_i \mid \theta_{-i}, x)$ for all $k = 1, \ldots, K$
(ii) Go to 1

2.1.2.3. Generate samples using the Chinese Restaurant Process. The distribution is integrated over π to find z_i conditional to $\{z_{1:i-1}\}$

$$z_i \mid z_{1:i-1} \sim p(z_{ik} = 1 \mid z_{1:i-1}) = \frac{N_{k,-i} + \frac{\alpha}{K}}{i - 1 + \alpha} \qquad N_{k,-i} = \sum_{j \neq i} z_{jk} \tag{2.40}$$

When $K \to \infty$, this goes to

$$\begin{aligned} \lim_{K\to\infty} p(z_{ik} = 1 \mid z_{1:i-1}) &= \frac{N_{k,-i}}{i - 1 + \alpha} \quad \text{, if } N_{k,-i} > 0 \\ \lim_{K\to\infty} \sum_{k|N_{k,-i}=0} p(z_{ik} = 1 \mid z_{1:i-1}) &= \frac{\alpha}{i - 1 + \alpha} \quad \text{, for all } N_{k,-i} = 0 \end{aligned} \tag{2.41}$$

A new sample is assigned to a mixture component with probability proportionate to the population of that mixture component. A new mixture component is created for this sample with probability α. Samples are generated by the following steps:

(i) Assign first sample to first mixture component $z_{11} = 1$
(ii) Initialize $i = 2$
(iii) Pick $z_{ik} = 1$ with probability $N_{k,-i} = \sum_{j \neq i} z_{jk}$, or
pick an unused assignment z_i with probability α
(iv) Increment i, go to step 3

2.1.2.4. Gibbs sampling based on the Chinese Restaurant Process. For applying Gibbs sampling, full conditionals for $\tilde{\theta}_k$ and z_i are needed. Full conditional for $\tilde{\theta}_k$ is the same as in the finite mixture model:

$$\begin{aligned} p(\tilde{\theta}_k \mid x, z) &= \mathcal{N}(\tilde{\theta}_k \mid m_k, V_k) \\ V_k &= \frac{Q}{N_k + Q/P} \qquad m_k = \frac{S_k}{N_k + Q/P} \end{aligned} \tag{2.42}$$

Full conditional for z_i is obtained as follows:

$$p(z_{ik} = 1 \mid z_{-i}, x_i, \tilde{\theta}) \propto p(x_i \mid z_i, \tilde{\theta}_k)\, p(z_{ik} = 1 \mid z_{-i})$$

$$p(z_{ik} = 1 \mid z_{-i}, x_i, \tilde{\theta}) \propto F(x_i|\tilde{\theta}_k)\, \frac{N_{k,-i} + \frac{\alpha}{K}}{i - 1 + \alpha} \tag{2.43}$$

Probability to choose a previous $\tilde{\theta}_k$:

$$\lim_{K\to\infty} p(z_{ik} = 1 \mid z_{-i}, x_i, \tilde{\theta}_k) \propto F(x_i|\tilde{\theta}_k)\, N_{k,-i} \quad \text{, if } N_{k,-i} > 0 \tag{2.44}$$

Pick a new $\tilde{\theta}_k$ from $P_i = p(\theta|x_i) = \mathcal{N}(\theta|\frac{Q}{1+Q/P}, \frac{x_i}{1+Q/P})$:

$$\lim_{K\to\infty} \sum_{k|N_{k,-i}=0} p(z_{ik} = 1 \mid z_{-i}, x_i, \tilde{\theta}_k) \propto \alpha \int F(x_i|\theta)H(\theta)d\theta$$

$$\propto \alpha\, \mathcal{N}(x_i \mid 0, Q + P) \tag{2.45}$$

Gibbs sampling algorithm for Chinese Restaurant Process is as follows:

(i) For $i = 1, \ldots, N$, let $z_{ik} = 1$
- If $N_{k,-i} = 1$, remove $\tilde{\theta}_k$
- Sample z_i from $p(z_i \mid z_{-i}, x_i, \tilde{\theta})$.
- If a new cluster is created, sample a new $\tilde{\theta}_k$ from H

(ii) Sample $\tilde{\theta}_k$ from $p(\tilde{\theta}_k|x, z)$ for all $k = 1, \ldots, K$

(iii) Go to 1

2.1.2.5. Collapsed Gibbs sampling based on the Chinese Restaurant Process. The distribution is integrated over $\tilde{\theta}$ before deriving the conditional over z_i.

$$\begin{aligned}
\text{Pick a previous } \tilde{\theta}_k \text{ with prob. } &\propto N_{k,-i} \int F(x_i|\theta) P_{-i}(\theta) d\theta \\
&\propto N_{k,-i}\, \mathcal{N}(x_i \mid m_k, Q + V_k) \\
\text{Pick a new } \tilde{\theta}_k \text{ from } P_i \text{ with prob. } &\propto \alpha \int F(x_i|\theta) H(\theta) d\theta \\
&\propto \alpha\, \mathcal{N}(x_i \mid 0, Q + P) \qquad (2.46)
\end{aligned}$$

where P_{-i} is the posterior of θ given all x_{-i}:

$$\begin{aligned}
P_{-i}(\theta) &= \mathcal{N}(\theta \mid m_k, V_k) \\
V_k = &\frac{Q}{N_{k,-i} + Q/P} \qquad m_k = \frac{S_{k,-i}}{N_{k,-i} + Q/P} \qquad (2.47)
\end{aligned}$$

Collapsed Gibbs sampling for Chinese Restaurant Process is as follows:

- Sample z_i from $p(z_i|z_{-i}, x)$ for all $i = 1, \ldots, N$
- Go to 1

3. INFINITE MULTIWAY MIXTURE

In this chapter, we describe an infinite multiway mixture (IMM) model that we developed by extending the infinite mixture model to $D+\Delta$ dimensions. Its parameters are represented as a tensor factorization. We define a D-way Poisson mixture, where a large observed tensor X is generated by the mixture proportions π_d and a smaller latent tensor Θ, which is represented as a factorization of M latent factors Θ_m of varying dimensionalities.

3.1. Multiway clustering

Multiway clustering is a problem that has recently gained attention. In [13], the multiway clustering problem is formulated with reference to earlier work on using hypergraphs to approach VLSI and PCB clustering placement problem. As elaborated in [14], a hypergraph is a general representation that contains hyperedges of any size that relate any combination of entities. In [15], a general multiway framework is presented to handle various kinds of hyperedges. Our model assigns D-tuples of objects to D-tuples of clusters, thus only involves D-way hyperedges that relate D objects of different types.

We use D-way tensors to represent the variables. In [16], a probabilistic tensor factorization framework is presented for multiway analysis. We use a similar framework to represent the latent D-way tensor of component parameters in the multiway mixture. In [17], an MCMC method was proposed for nonparametric biclustering problem. Biclustering is a special case of D-way clustering for $D = 2$, thus its solution can be applied to the general multiway problem.

In the following sections, we first formulate the multiway clustering problem as a finite mixture, and present a variational inference method. Then, we suggest an MCMC inference method to be used with the infinite mixture model.

3.2. The D-way Poisson mixture model

In our problem, we have D types of objects, and N_d objects from each object type $d \in \{1, \ldots, D\}$. Each of the observations is given for a D-tuple of these objects, together making up the D-way tensor X with sizes $N_1, \ldots, N_D$ in its D dimensions. We model X as a Poisson mixture of latent parameters in a smaller D-way tensor Θ with sizes $K_1, \ldots, K_D$. Here, K_d is the number of clusters for a given dimension, and in general it is significantly smaller than N_d. A D-way tensor is indexed by an 'index set' of D indices. Each observation in the tensor X is denoted $X(\iota)$ where ι is the index set $\{i_1, \ldots, i_D\}$ and $i_d \in \{1, \ldots, N_d\}$. Similarly, the tensor Θ is indexed by κ that is the index set $\{k_1, \ldots, k_D\}$ where $k_d \in \{1, \ldots, K_d\}$.

3.3. Layer assignments

In D-way clustering, for each object type d, N_d objects of this type are to be assigned to the corresponding K_d clusters. For $D = 1$, single observations in X are assigned to single parameters in Θ. For $D = 2$, rows of observations in X are assigned to rows of parameters in Θ, and columns to columns. When $D = 3$, matrices in three different orientations from X are assigned to matrices of corresponding orientations in Θ. For the general case, (D-1)-way tensors in X are assigned to (D-1)-way tensors in Θ. We call such a (D-1)-way tensor a 'layer', and indicate it by a dimension d, and a value for its index j_d. The layer's orientation is given by d, and its placement inside the tensor by j_d. We call C_d an indicator variable. For each orientation d, that $C_d(i_d, k_d) = 1$ indicates that the layer at i_d of X is assigned to the layer at k_d of Θ. A single observation $X(\iota)$ is thus assigned by the indicators $C_1, \ldots, C_D$ to the layers at the indices $k_1, \ldots, k_D$ of Θ. These indices form the set κ, and as a result, the observation is assigned to the latent parameter at $\Theta(\kappa)$. In the mixture model, we assume that each observation $X(\iota)$ is Poisson distributed with the intensity as the latent parameter $\Theta(\kappa)$ to which it is assigned.

$$X(\iota) \mid \Theta, C \sim \prod_{\kappa} \mathcal{PO}(\Theta(\kappa))^{\prod_{d=1}^{D} C_d(i_d, k_d)} \tag{3.1}$$

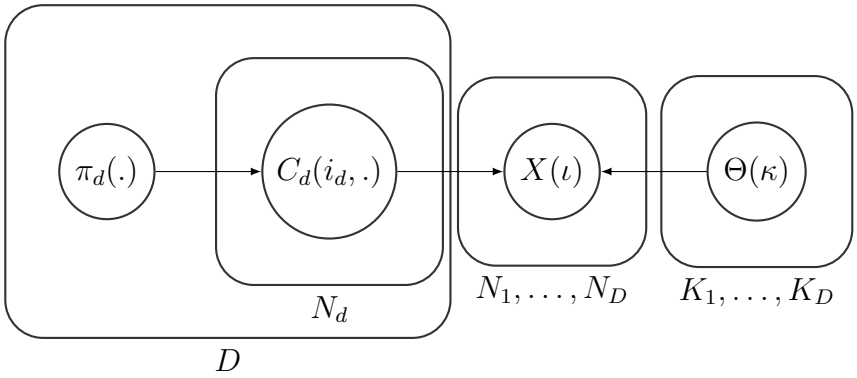


Figure 3.1. The D-way mixture model

We model each of the vectors $C_d(i_d, .)$ by a discrete distribution $\pi_d(.)$ of size K_d. This vector is in turn modeled by a symmetric Dirichlet prior with concentration α_d. The full model is shown in Figure 3.1.

$$\begin{aligned} C_d(i_d, .) \mid \pi_d &\sim Discrete(\pi_d(.)) \\ \pi_d(.) &\sim Dir(\frac{\alpha_d}{K_d}, \ldots, \frac{\alpha_d}{K_d}) \end{aligned} \tag{3.2}$$

3.3.1. Representing the latent tensor

Up to this point, we have described a general D-way mixture model. What makes our model specific is the representation of the latent tensor Θ. We assume that Θ is a function of other M latent factors Θ_m of different dimensionalities.

$$\Theta(\kappa) = \sum_{\beta} \prod_{m=1}^{M} \Theta_m(\gamma_m) \tag{3.3}$$

The factorization is summed over a set of indices β to get the latent tensor Θ indexed by κ. Here, β denotes an 'additional' set of indices $\{k_{D+1}, \ldots, k_{D+\Delta}\}$ that extends the 'original' set denoted by κ. The union of these two gives the full set of indices $\kappa \dot{\cup} \beta = \{k_1, \ldots, k_{D+\Delta}\}$. Each of the M factors is indexed by γ_m, such that $\cup_{m=1}^{M} \gamma_m = \kappa \dot{\cup} \beta$. We consider the factors Θ_m as the actual hidden parameters, and put on them a Gamma

prior, which is conjugate with the Poisson distribution.

$$\Theta_m(\gamma_m) \sim \mathcal{G}(A, \frac{B}{A}) \tag{3.4}$$

3.3.2. Partitioning a factor's indices

The index set γ_m of a factor is partitioned into three disjoint sets in two consecutive steps as follows:

$$\gamma_m = \eta_m \dot{\cup} \beta_m = \eta_m \dot{\cup} \sigma_m \dot{\cup} \lambda_m \tag{3.5}$$

In the first step, it is partitioned into its 'original' indices $\eta_m = \gamma_m \cap \kappa$ and 'additional' indices $\beta_m = \gamma_m \cap \beta$. We then introduce $\beta_{-m} = \cup_{m' \neq m} \beta_m$ to denote the additional indices that belong to any factor other than m. In the second step, β_m is further partitioned into the indices shared with other factors $\sigma_m = \beta_m \cap \beta_{-m}$ and the indices that are not shared $\lambda_m = \beta_m \backslash \beta_{-m}$.

3.3.3. Determining the parameters of the model

For a given D, a finite multiway mixture model is selected by determining:

(i) The sizes $\{K_1, \ldots, K_D\}$ of the latent tensor Θ (the number of clusters for all object types).
(ii) The concentration parameters $\alpha_1, \ldots, \alpha_D$ for each of the object types.
(iii) The number of factors M, and the index sets $\gamma_1, \ldots, \gamma_M$ for each of these factors. When these are given, we can also determine the set of additional indices $\beta = \cup_{m=1}^{M} \gamma_m \backslash \kappa$.
(iv) The prior parameters A and B for the factors Θ_m.

Various factorizations of Θ lead to different models. To mention two basic examples: When $M = 1$ and $\gamma_1 = \kappa$, the latent tensor $\Theta(\kappa)$ is modelled directly. When $M = D$

and $\gamma_d = \{k_d\}$, the tensor Θ is the product of D vectors $\Theta_d(k_d)$.

3.4. Variational inference for the finite mixture

We develop an Expectation-Maximization algorithm that involves these steps:

(i) Calculate the expectation of $p(C \mid X, \Theta, \pi)$.
(ii) For each $d \in \{1, \ldots, D\}$, calculate the π_d that maximizes $p(X, C, \Theta, \pi)$.
(iii) For each $m \in \{1, \ldots, M\}$, calculate the Θ_m that maximizes $p(X, C, \Theta, \pi)$.

In the expectation step (1), we use the posterior of the layer assignments.

$$p(C \mid X, \Theta, \pi) \propto \{\prod_{\iota} \prod_{\kappa} \mathcal{PO}(X(\iota) \mid \sum_{\beta} \prod_{m=1}^{M} \Theta_m(\gamma_m))^{\prod_d C_d(i_d, k_d)}\}$$
$$\{\prod_{d} \prod_{i_d} \prod_{k_d} \pi_d(k_d)^{C_d(i_d, k_d)}\} \tag{3.6}$$

In the maximization step (2), we update $\pi_d(k_d)$ by the equation:

$$\pi_d^*(k_d) = \frac{\frac{\alpha_d}{K_d} - 1 + \sum_{i_d} \mathbb{E}[C_d(i_d, k_d)]}{\alpha_d - K_d + N_d} \tag{3.7}$$

In the next step (3), we update $\Theta_m(\gamma_m)$ by the following formula:

$$\Theta_m^*(\gamma_m) = \frac{A - 1 + \sum_{\iota} \sum_{\kappa \setminus \gamma_m} \{X(\iota) \frac{\prod_{d: k_d \in \lambda_m} K_d}{\sum_{\lambda'_m \neq \lambda_m} \Theta_m(\eta_m \cup \sigma_m \cup \lambda'_m)}\} \mathbb{E}[\prod_d C_d(i_d, k_d)]}{\frac{B}{A} + \sum_{\iota} \sum_{\kappa \setminus \gamma_m} \{\sum_{\beta} \prod_{m' \neq m} \Theta_{m'}(\gamma_{m'})\} \mathbb{E}[\prod_d C_d(i_d, k_d)]} \tag{3.8}$$

For any factor $\Theta_m(\gamma_m)$ with no additional indices (for each m where $\beta_m = \emptyset$), the fraction in the numerator of the formula reduces to 1. When there is no factorization ($M = 1$), the coefficient in the summation in the denominator also reduces to 1.

3.5. The infinite mixture and MCMC inference

By taking a finite D-way mixture, and bringing $K_d \to \infty$ for some or all of the object types, we can obtain an infinite multiway mixture (IMM). We are developing an MCMC method for inferring the layer assignments in such a nonparametric multiway mixture model. A variety of MCMC methods for DPM are presented in [18] including Gibbs sampling, Metropolis-Hastings updates and auxiliary parameters to handle both conjugate and non-conjugate priors. In [17], such a method is developed for a nonparametric biclustering model, which can be obtained from our D-way model for $D = 2$. The method proposed is based on a property which we will express in our terms as follows. When C_d are given, for each κ, there is a set or a 'block' of observations that are assigned to it:

$$\{X(\iota) : \kappa\} \;=\; \{X(\iota) : \prod_d C_d(i_d, k_d) = 1\} \tag{3.9}$$

Conditional to Θ, these blocks of observations are independent and thereby their likelihoods:

$$p(X \mid \Theta, C) \;\sim\; \prod_\kappa p(\{X(\iota) : \kappa\} \mid \Theta(\kappa)) \tag{3.10}$$

In case of a conjugate prior, we can also integrate out Θ to get the following:

$$p(X \mid C) \;\sim\; \prod_\kappa \int p(\{X(\iota) : \kappa\} \mid \Theta(\kappa))\; p(\Theta(\kappa))\; d\Theta(\kappa) \;=\; \prod_\kappa p(\{X(\iota) : \kappa\}) \tag{3.11}$$

Using this property, an MCMC algorithm similar to [17] can be developed for the infinite D-way mixture. IMM is a simple but powerful generative model that combines infinite mixture modeling with tensor factorization. The index notation borrowed from tensor factorization is used to formulate a multiway mixture model over several dimensions. IMM provides an elegant model that explores certain theoretical ideas from related domains.

4. INFINITE MIXTURE OF PIECEWISE LINEAR SEQUENCES

In this chapter, a Bayesian nonparametric method for clustering time sequences is presented, based on the assumption that samples in each cluster are distributed around a number of line segments. The model is convenient for analysing results from gene expression experiments that involve a number of temporal 'phases' of collected data. An infinite mixture model with dynamic number of clusters and dynamic hyperparameters is designed. As its components, piecewise linear sequences were used to model the 'trends' of behavior in these phases. A Markov chain Monte Carlo scheme was developed to sample from the mixture's posterior. The presented method is applied to a dataset of gene expression profiles with six phases of behaviour. Infinite mixture of piecewise linear sequences (IMPLS) is an automatic model-based method that yields a summarized representation of the given set of gene expression profiles.

4.1. Introduction

Clustering of time-series data have been useful in various biological problems, the most popular being gene expression profiles. As was pointed out by Yeung *et al.* [19], algorithms used to cluster gene expression profiles are in practice mostly heuristic-based, as opposed to more rigorous model-based approaches. These heuristic methods such as hierarchical clustering [20], self-organizing maps [21] and k-means [22] are practical and fast methods for extracting further observations from a dataset at hand, but their results are not dependable, because they require users to make decisions that are not based on a theoretic model. In the model-based approach, decisions only involve the model being constructed and inference methods can be developed independently from the model. Moreover, a proper model-based approach can be combined with heuristic methods, which are only to be employed before the time of inference to help with model decisions, and after the time of inference for further inspecting the results.

In the last decade, nonparametric Bayesian models [2] emerged as a model-based option that allows enhanced flexibility in modeling structural relations. In our study, a nonparametric Bayesian model is employed for time-series analysis. The most well-known nonparametric Bayesian models are infinite mixtures. Infinite mixtures involve stochastic processes as priors. Dirichlet process (DP) and Pitman-Yor process (PYP) are such priors. These priors have conditional probabilistic 'constructions' like Chinese restaurant processes (CRP) or stick breaking processes, and these constructions allow for deriving iterative inference methods [18].

Medvedovic *et al.* [23] applied Gaussian infinite mixture model for clustering gene expressions. An MCMC inference method was applied by Qin [24] based on collapsed Gibbs sampling. In this model, based on a CRP construction of a DP mixture, the set of samples in a given gene expression profile are assumed to be independently distributed. Joshi *et al.* [25] extended this approach for the simultaneous co-clustering of genes and samples.

Our model differs from these models in that we model each mixture component with a corresponding piecewise linear sequence (PLS). In a PLS, samples are divided into line segments that designate the 'phases' of an experiment. It is assumed that every phase of samples displays a linear relation with a constant slope, and there are jumps in between these segments. These slopes can be interpreted as different 'trends' of gene expression behavior. We model gene expressions as an infinite mixture of PLS (IMPLS), summarizing the dataset in terms of these trends of behavior. The MCMC inference method samples most of the hyperparameters, reaching convergence in a larger parameter space. The idea of the PLS model derives from experimental observations by Dikicioğlu *et al.* [26]. In their experiments with the yeast cell, they discovered that gene expressions that come from time points of the same order (seconds, minutes, hours) showed correlated behavior among themselves. Neighboring time points naturally organized into segments that mark the 'phases' of their experiment. To apply IMPLS to another dataset, its common segment structure has to be decided by making a similar preliminary correlation analysis of samples from different time points.

4.2. The model

Suppose that we have N genes indexed by $i \in \{1, \ldots, N\}$ and their expression profiles x_i, vectors of size M, which are to be modelled as distributed around an unknown number of piecewise linear sequences. Cluster assignments z_i of these genes are assumed to come from a two-parameter CRP, an iterative construction for a Pitman-Yor process:

$$z_{1:N} \mid \alpha, d \ \sim CRP(\alpha, d) \tag{4.1}$$

A PLS is defined by L parameters in the following order: *initial value, slope of first segment, jump to second segment, slope of the second segment, jump to third segment,* etc. and the prior variances of these three types of parameters are given by $V_{\text{init}}, V_{\text{jump}}, V_{\text{slope}}$. These variances form the diagonal of the matrix Σ. For every cluster $k \in \{1, \ldots, K\}$ there is an L-vector μ_k that defines a PLS with a Gaussian prior:

$$\mu_k \mid V_{\text{init}}, V_{\text{jump}}, V_{\text{slope}} \ \sim \mathcal{N}(\mu_k \mid 0, \Sigma) \tag{4.2}$$

Each cluster also has a precision (inverse variance) parameter λ_k with a Gamma prior:

$$\lambda_k \mid a, b \ \sim \mathcal{G}(\lambda_k \mid a, b) \tag{4.3}$$

Finally, we have the likelihood, which determines that each observation is distributed according to a Gaussian with mean $C\mu_k$ and variance $1/\lambda_k$, where k is the cluster that this sample belongs to. C is a constant matrix that transforms PLS parameters into their corresponding PLS mean:

$$x_i \mid \mu, \lambda, z_i \ \sim \prod_{k=1}^{K} \mathcal{N}(x_i \mid C\mu_k, \lambda_k^{-1} I)^{z_{ik}} \tag{4.4}$$

4.3. MCMC inference

In Markov chain Monte Carlo (MCMC) methods, a Markov chain is designed to converge to a specified target distribution. Gibbs sampling and collapsed Gibbs sampling are MCMC methods that are frequently used. Metropolis-Hastings (MH) is a more generic MCMC method based on rejection sampling [27]. Qin [24] employed collapsed Gibbs sampling to gene expressions by integrating out cluster centers and variances. But this is not possible with our model. We run MH steps to sample cluster precisions λ_k, and use these values to run collapsed Gibbs sampling steps to sample z_i by integrating out the cluster centers μ_k. Our MCMC algorithm consists of three steps repeatedly applied to converge to the target distribution $p(x, z, \lambda, \alpha, d, a, b)$

(i) For each $k = 1, \ldots, K$, apply MH steps to re-sample λ_k by $p(\lambda_k \mid x_{1:N}, z_{1:N})$.

(ii) For each $i = 1, \ldots, N$, apply collapsed Gibbs sampling for z_i by $p(z_i \mid x_{1:N}, z_{-i}, \lambda_{1:K})$ using auxiliary variable method for sampling new λ_k.

(iii) MH steps to sample α, d, a, b by their respective uninformative priors $\frac{1}{\alpha}$, $\frac{1}{d}$, 1 and b.

The PLS prior parameters V_{init}, V_{jump}, V_{slope} are fixed at a sufficiently large number to assign equal probability for different PLS parameter values.

4.4. Likelihoods and linear segments

The samples to be partitioned are time series vectors of size M. Likelihood of the mixture component parameters are given by $\mathcal{N}(\mu_k \mid 0, \Sigma)$. Likelihood of the observations is given by $\mathcal{N}(x_i \mid C\ \mu_k, \lambda^{-1} I)$.

The parameter μ_k determined by the first distribution is a vector of size $L < M$, and the covariance of this distribution Σ is a non-negative diagonal matrix of size $L \times L$. In the second line, μ_k is transformed by the C matrix of size $M \times L$. In order for the vector that results from this transformation to be a piecewise linear sequence, C matrix must be constructed as below.

Let's assume that there are J linear segments and that the segment j on vector x_i begins from the index m_j and goes until the index $m_{j+1} - 1$. In this case, $m_1 = 1$ and $m_J \leq M$. Each linear segment requires two parameters: the amount of jump at the first element of the segment and the amount of constant change from the second element onwards. For instance if we want to segment $M = 7$ into two segments of size 3 and 4, we must choose $J = 2, m_1 = 1, m_2 = 4$.

In order to construct the C matrix we first need to determine the r vector:

$$\begin{aligned} r_1 &= 1 \\ r_m &= r_{m-1} + \begin{cases} 1 & \text{if } \exists j(m = m_j \vee m = m_j + 1) \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{4.5}$$

This vector is a numerical sequence that begins with 1 and is incremented at the elements that correspond to the first and second elements of the linear segments. The value r_m gives the number of parameters required for the first m observations, therefore the total number of parameters is $r_M = L$. For the example above it will be determined to be $r = (1, 2, 2, 3, 4, 4, 4)$. The matrices Σ and C are determined as follows:

$$\Sigma_{l,l} = \begin{cases} P_0 & l = 1 \text{ ise} \\ V_{\text{slope}} & \text{if } \exists m, j(m = m_j \wedge r_m = l) \\ V_{\text{jump}} & \text{otherwise} \end{cases}$$

$$\begin{aligned} C_{1,1:L} &= (1, 0, 0, \ldots, 0) \\ C_{m,l} &= C_{m-1,l} + \begin{cases} 1 & \text{if } r_m = l \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{4.6}$$

Among the values that constitute the diagonal of Σ, V_{init} is the initial variance of the first linear segment, V_{slope} is the variance of the jumps between segments, and V_{jump} is the variance of the constant change in the subsequent elements of a linear segment. Each row of the C matrix is obtained by inserting one of the parameters

to the previous row. The column whose values are incremented, corresponds either to the jump parameter if it passes from one linear segment to another, or to the constant change parameter if the same segment continues. For the example above Σ and C will be as below.

$$\Sigma = diag(V_{\text{init}}, V_{\text{jump}}, V_{\text{slope}}, V_{\text{jump}})$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 1 & 2 & 1 & 1 \\ 1 & 2 & 1 & 2 \\ 1 & 2 & 1 & 3 \end{bmatrix} \tag{4.7}$$

4.5. Posterior probabilities

Chinese restaurant process, with the likelihoods, constitutes an infinite mixture model. While making inference, we need posterior probability values in order to assign the observations to the mixture components. These probabilities, for each assignment, are conditional to the previous assignments, just like the prior probabilities:

$$p(z_{ik} = 1 \mid z_{i-1}, x_i, \theta_k) \propto$$

$$\begin{cases} N_k \; F(x_i \mid \theta_k) & \text{if } k \leq K_{i-1}^{+} \\ \alpha \int F(x_i \mid \theta) \; p(\theta) d\theta & \text{otherwise} \end{cases} \tag{4.8}$$

The probability to be inserted to a previous mixture component, is multiplied by the likelihood of that mixture component. Whereas the probability to open a new mixture component, is multiplied by the expected value according to the prior distribution of the likelihood parameter.

Since we will apply collapsed Gibbs sampling, by summating over θ_k, we obtain the following posterior probabilities:

$$p(z_{ik} = 1 \mid z_{i-1}, x_i) \ \propto$$

$$\begin{cases} N_k \ \int F(x_i \mid \theta) \ p(\theta \mid x_{1:i-1}, z_{1:i-1}) d\theta & \text{if } k \leq K_{i-1}^+ \\ \alpha \int F(x_i \mid \theta) \ p(\theta) d\theta & \text{otherwise} \end{cases} \tag{4.9}$$

By writing model equations in place we obtain the following equation:

$$p(z_{ik} = 1 \mid z_{i-1}, x_i) \ \propto$$

$$\begin{cases} N_k \ \exp(\psi_0 + \psi_3^k - \frac{1}{2Q} x_i^T x_i \\ \quad + S_k^T \psi_4^k S_k + (S_k + x_i)^T \psi_5^k (S_k + x_i)) & \text{if } k \leq K_{i-1}^+ \\ \alpha \exp(\psi_0 + \psi_1 - \frac{1}{2Q} x_i^T x_i + x_i^T \psi_2 x_i) & \text{otherwise} \end{cases} \tag{4.10}$$

Here the vector S_k is the sum of the observations in mixture component k at the moment after the observation $i-1$ is assigned. The statistics designated by ψ are as follows:

$$\begin{aligned} \psi_0 &= -\frac{M}{2} \log 2\pi - \frac{1}{2} \log |QI| \\ \psi_1 &= -\frac{1}{2} \log |\Sigma| + \frac{1}{2} \log |V_1| \\ \psi_2 &= \frac{1}{2Q^2} C V_1 C^T \\ \psi_3^k &= -\frac{1}{2} \log |V_{N_k}| + \frac{1}{2} \log |V_{N_k+1}| \\ \psi_4^k &= -\frac{1}{2Q^2} C V_{N_k} C^T \\ \psi_5^k &= \frac{1}{2Q^2} C V_{N_k+1} C^T \\ V_n &= (\Sigma^{-1} + \frac{n}{Q} C^T C)^{-1} \end{aligned} \tag{4.11}$$

The matrices V_1, V_{N_k} and V_{N_k+1} are determined by the equation for V_n.

4.6. Experiment and results

We have applied our proposed method on the carbon dataset by Dikicioğlu *et al.* [26]. They collected 5667 transcriptomic profiles of the yeast cell (*S. cerevisiae*) in response to the sudden and transient removal of either carbon or nitrogen limitation. For each gene, samples were collected at 20 sec, 40 sec, 60 sec, 8 min, 16 min, 24 min, 32 min, 1 hr, 2 hr, 3 hr, 4 hr, 5 hr, 7 hr, 1st and 2nd steady states. According to their analysis, individual time points in the dynamic scale arranged into clusters forming distinct phases in which the transcriptome response was observed to be similar. We use this result to determine the C matrix in our PLS model by assuming a piecewise linear sequence structure with 6 phases for this dataset. The MCMC was run for 10.000 iterations, sampling the assignments $z_{1:N}$, cluster precisions λ_k as well as the hyperparameters α, d, a, b. After the inference has finished, the most probable iteration was chosen as the iteration 3540, which contains 269 clusters. Example clusters from

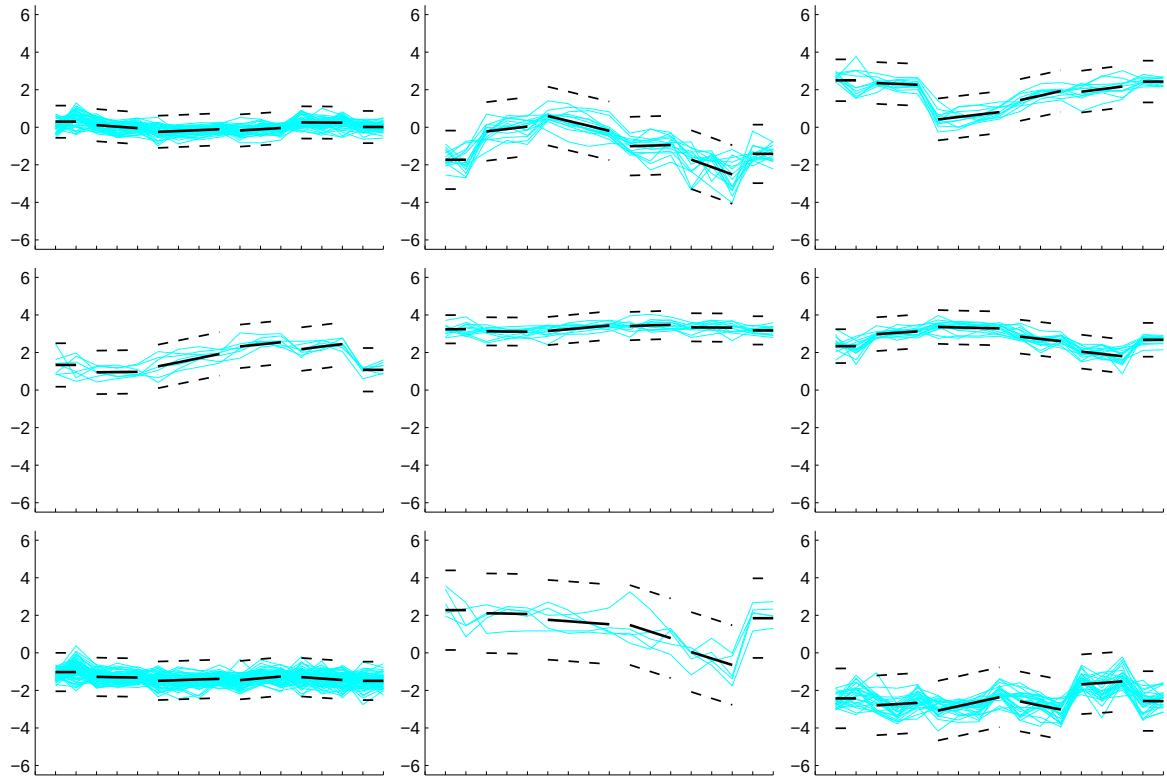


Figure 4.1. Random clusters from the iteration with maximum probability in the experiment. Black lines indicate the PLS mean $C\mu_k$ and the range $\pm 3\lambda_k^{-\frac{1}{2}}$. Blue lines indicate the gene expression profiles in that cluster.

this iteration are shown on Figure 4.1. The calculated distributions of some parameters can be observed from the histograms of the sampled parameters on Figure 4.2. The number of clusters K, the cluster precisions λ_k and the hyperparameters α, d, a, b all converge according to the target distribution $p(x, z, \lambda, \alpha, d, a, b)$ and oscillate around their respective marginal posteriors.

The hyperparameters a, b of the cluster precisions λ_k converge to their respective distributions around 8 and 1. The other two hyperparameters α, d that determine the nonparametric prior's tendency to create more clusters, oscillate around 60 and 0, whereas the cluster count K oscillates around the range 270-285. In the most probable iteration 3540 with 269 clusters, the cluster precisions λ_k are observed to be distributed around 0,2. Though the convergence results of our method are desirable, the clusters are still very numerous, and as you can see on Figure 4.1, some clusters can be very similar in some of their phases of behavior. As actual gene expression profiles are not naturally organized into clearly separated clusters of different behavior, the 'clusters'

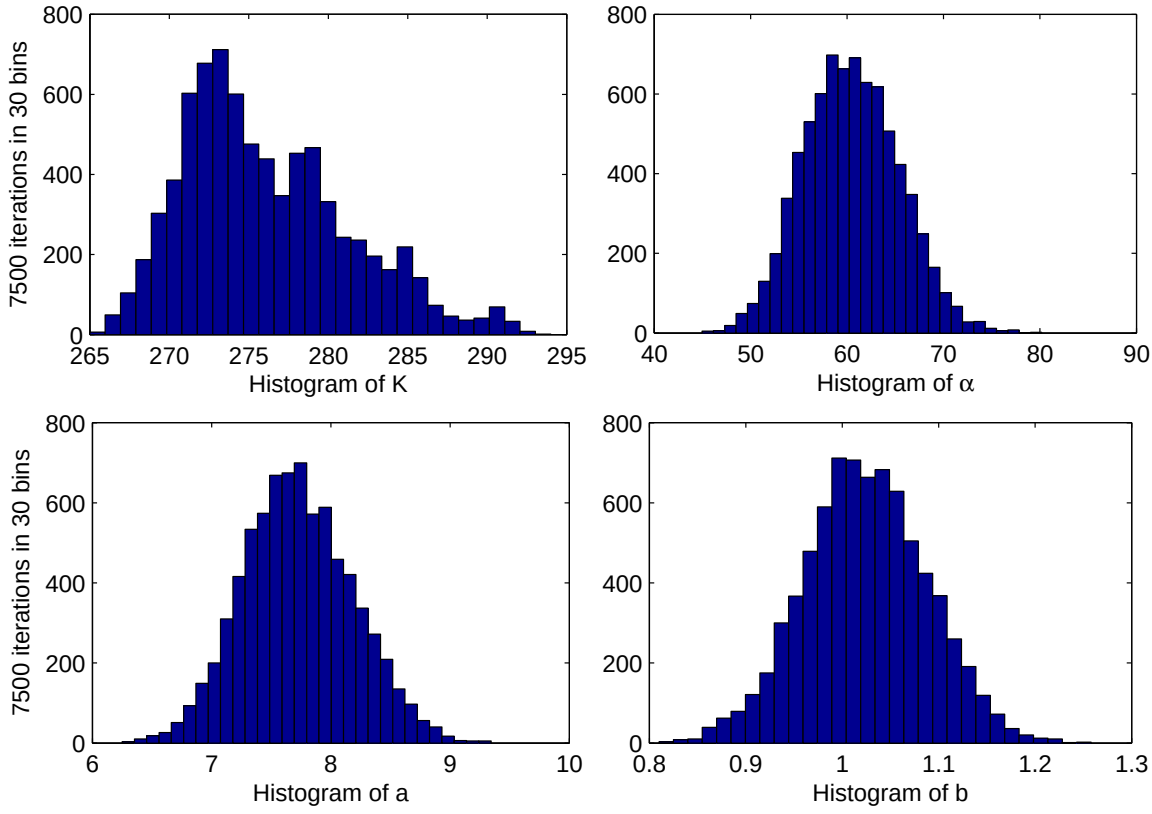


Figure 4.2. Histograms of K, α, a, b through the iterations.

resulting from our model-based inference are not also clearly separated and they show significant similarities in their various phases.

Inference results of the IMPLS can be interpreted as a model-based 'slicing' of the larger dataset of gene expression profiles into a smaller representative dataset, whose elements are groups of genes with a corresponding piecewise linear sequence that show their 'trends' of behavior in the given 'phases' of the experiment. Thus, the set of 269 clusters provide a summarized representation of the larger dataset of 5667 expressions, faithfully to our PLS mixture assumptions. However, the clustering results also invite further analysis before designing hypothesis testing schemes. It would be helpful to inspect similarity relationships among the resulting clusters, as well as to calculate posteriors for individual clusters with respect to the target distribution by MCMC.

5. CUMULATIVE STATISTICS AND ENTROPY AGGLOMERATION

In this chapter, we present our methodology of *cumulative statistics* [28]. It is a statistical approach that represents sample sets of partitionings and feature allocations by a cumulative formulation. We also describe a clustering algorithm called Entropy Agglomeration (EA). The use of EA will be demonstrated in the next chapter. Cumulative statistics logically lies outside of Bayesian methodology, but it originates from a theoretical question that emerges in the context of infinite mixture models, which are among the prominent models of Bayesian nonparametrics. Here we give an account of nonparametric priors used for infinite mixture models in order to express the theoretical question required us to develop this new approach, cumulative statistics.

5.1. Partitioning of elements in Bayesian nonparametrics

The most well-studied nonparametric prior that is used in infinite mixture models is *Dirichlet process* [1, 2] that makes its decisions according to a *rich-gets-richer* rule, which makes new elements more likely to be assigned to the mixture components with more numerous elements. A well-studied superclass of DP is *Poisson-Dirichlet process* (PDP) [29, 30] that is based on a modified *rich-gets-less-richer* rule, which allows it more likely to create new mixture components proportionally to the current number of mixture components. The advantage of DP and PDP is that they are well-known concepts as part of a greater literature of stochastic processes and are known to have many theoretical properties and relations with other mathematical objects. The disadvantage of these formulations is that they model a joint distribution over two different aspects of an infinite mixture configuration —element assignments and mixture component parameters— unable to distinguish clearly between these two tasks. As we are more interested in the first aspect —element assignments— since they provide the clustering solution, we need to decouple it from the second aspect —mixture component parameters. This is achieved by two basic incremental constructions for

infinite mixture models: *Chinese restaurant process* (CRP) [7] formulates an infinite mixture model by generating a sequence of conditional element assignment probabilities, whereas *stick-breaking process* (SBP) [31] formulates an infinite mixture model by generating a sequence of unconditional element assignment probabilities. CRP and SBP constructions are especially helpful in developing computational strategies for probabilistic inference. Since the two aspects of an infinite mixture configuration may live in different kinds of spaces, they pose different kinds of computational difficulties: element assignments always live in a combinatorial space, whereas mixture component parameters may live either in a combinatorial space or a continuous space depending on the model. CRP and SBP allows us to divide this combined problem into simpler sub-problems to develop more effective approaches, the most well-known methods being Markov chain Monte Carlo (MCMC) algorithms [18].

The conceptual and computational conveniences afforded by these formulations motivated further theoretical and practical developments: [32] combined two CRPs in parallel to obtain an efficient probabilistic method for analysing dyadic data; [33] connected several SBPs serially to obtain the *hierarchical Dirichlet process*. These developments also inspired the invention of construction schemes for different nonparametric models, such as *Indian buffet process* (IBP) [34] for infinite feature models [35] and the *fragmentation-coagulation process* for analysing sequence data [36]. In this way, based on a small variety of simple nonparametric priors, a large body of research has grown in the recent decades, commonly referred to as *Bayesian nonparametrics* [37]. Element assignments of an infinite mixture configuration can be represented by a partitioning, i.e. a set of blocks in which each element appears once. As a result, to infer element assignments of an infinite mixture model is equivalent to infer a *random partitioning*. This allows us to use certain sampling methods to obtain a sample set of partitionings drawn from the infinite mixture posterior. To apply collapsed Gibbs sampling on an infinite mixture model, its mixture component parameters —treated as irrelevant (nuisance) parameters— have to be integrated out of the joint probability distribution of its mixture configuration. The resulting equation can then be used to sample its element assignments repeatedly in a closed loop. After several iterations the obtained sample set of partitionings will approximately represent the infinite mixture

posterior. If the posterior is peaked around a single partitioning, then the *maximum a posteriori* solution will be quite informative. However, in some cases the posterior is more diffuse and one needs to extract statistical information from the whole sample set of partitionings. This problem to 'summarize' the samples from the infinite mixture posterior was raised in bioinformatics literature in 2002 by Medvedovic and Sivaganesan for clustering gene expression profiles [23]. But the question proved difficult and they 'circumvented' it by using a heuristic linkage algorithm based on pairwise occurence probabilities [38, 39]. In this chapter, we approach this problem and propose basic methodology for summarizing sample sets of partitionings as well as feature allocations.

Nemenman *et al.* [40] showed that the entropy [41] of a DP posterior was strongly determined by its prior hyperparameters. Archer *et al.* [42] recently elaborated these results with respect to PDP. In other work, entropy was generalized to partitionings by interpreting partitionings as probability distributions [43, 44]. Therefore, entropy emerges as an important statistic for our problem, but new definitions will be needed for quantifying information in feature allocations. In the following sections, we define the problem and introduce *cumulative statistics* for representing partitionings and feature allocations. Then, we develop an interpretation for entropy function in terms of *per-element information* in order to quantify segmentation among their elements. Finally, we describe *entropy agglomeration* (EA) algorithm that generates dendrograms to summarize sample sets of partitionings and feature allocations. We demonstrate EA on infinite mixture posteriors for synthetic and real datasets as well as on a real dataset directly interpreted as a feature allocation.

5.2. Basic definitions and the motivating problem

We begin with basic definitions. A *partitioning* of a set of elements $[n] = \{1, 2, \ldots, n\}$ is a set of blocks $Z = \{B_1, \ldots, B_{|Z|}\}$ such that $B_i \subset [n]$ and $B_i \neq \emptyset$ for all $i \in \{1, \ldots, n\}$, $B_i \cap B_j = \emptyset$ for all $i \neq j$, and $\cup_i B_i = [n]$.[1] We write $Z \vdash [n]$ to designate

[1] We use the term 'partitioning' to indicate a 'set partition' as distinguished from an integer 'partition'.

that Z is a partitioning of $[n]$.[2] A sample set $E = \{Z^{(1)}, \ldots, Z^{(T)}\}$ from a distribution $\pi(Z)$ over partitionings is a multiset such that $Z^{(t)} \sim \pi(Z)$ for all $t \in \{1, \ldots, T\}$. We are required to extract information from this sample set. Our motivation is the following problem: a set of observed elements $(x_1, \ldots, x_n)$ are clustered by an infinite mixture model with parameters $\theta^{(k)}$ for each component k and mixture assignments $(z_1, \ldots, z_n)$ drawn from a two-parameter CRP prior with the concentration hyperparameter α and the discount hyperparameter d [7].

$$z \;\sim\; CRP(z; \alpha, d) \qquad \theta^{(k)} \;\sim\; p(\theta) \qquad x_i \mid z_i, \theta \;\sim\; F(x_i \mid \theta^{(z_i)}) \tag{5.1}$$

In the conjugate case, all $\theta^{(k)}$ can be integrated out to get $p(z_i \mid z_{-i}, x)$ to sample z_i [9]:

$$p(z_i \mid z_{-i}, x) \;\propto \int p(z, x, \theta)\; d\theta \;\propto \begin{cases} \frac{n_k - d}{n - 1 + \alpha} \int F(x_i|\theta)\; p(\theta|x_{-i}, z_{-i})\; d\theta & \text{if } k \leq K^+ \\ \frac{\alpha + dK^+}{n - 1 + \alpha} \int F(x_i|\theta)\; p(\theta)\; d\theta & \text{otherwise} \end{cases} \tag{5.2}$$

There are K^+ non-empty components and n_k elements in each component k. In each iteration, x_i will either be put into an existing component $k \leq K^+$ or it will be assigned to a new component. By sampling all z_i repeatedly, a sample set of assignments $z^{(t)}$ are obtained from the posterior $p(z \mid x) = \pi(Z)$. These $z^{(t)}$ are then represented by partitionings $Z^{(t)} \vdash [n]$. The induced sample set contains information regarding (1) CRP prior over partitioning structure given by the hyperparameters (α, d) and (2) integrals over θ that capture the relation among the observed elements $(x_1, \ldots, x_n)$. In addition, we aim to extract information from feature allocations, which constitute a superclass of partitionings [34]. A *feature allocation* of $[n]$ is a multiset of blocks $F = \{B_1, \ldots, B_{|F|}\}$ such that $B_i \subset [n]$ and $B_i \neq \emptyset$ for all $i \in \{1, \ldots, n\}$. A sample set $E = \{F^{(1)}, \ldots, F^{(T)}\}$ from a distribution $\pi(F)$ over feature allocations is a multiset such that $F^{(t)} \sim \pi(F)$ for all t. Current exposition will focus on partitionings, but we are also going to show how our statistics apply to feature allocations.

[2] The symbol '$\vdash$' is usually used for integer partitions, but here we use it for partitionings (=set partitions).

Assume that we have obtained a sample set E of partitionings. If it was obtained by sampling from an infinite mixture posterior, then its blocks $B \in Z^{(t)}$ correspond to the mixture components. Given a sample set E, we can approximate any statistic $f(Z)$ over $\pi(Z)$ by averaging it over the set E:

$$Z^{(1)}, \ldots, Z^{(T)} \ \sim \ \pi(Z) \qquad \Rightarrow \qquad \frac{1}{T} \sum_{t=1}^{T} f(Z^{(t)}) \ \approx \langle \ f(Z) \ \rangle_{\pi(Z)} \tag{5.3}$$

Which $f(Z)$ would be a useful statistic for Z? Three statistics commonly appear in the literature: First one is the *number of blocks* $|Z|$, which has been analyzed theoretically for various nonparametric priors [2, 7]. It is simple, general and exchangable with respect to the elements $[n]$, but it is not very informative about the distribution $\pi(Z)$ and therefore is not very useful in practice. A common statistic is *pairwise occurence*, which is used to extract information from infinite mixture posteriors in applications like bioinformatics [23]. For given pairs of elements $\{a, b\}$, it counts the number of blocks that contain these pairs, written $\sum_i [\{a, b\} \subset B_i]$. It is a very useful similarity measure, but it cannot express information regarding relations among three or more elements. Another statistic is *exact block size distribution* (referred to as 'multiplicities' in [35, 42]). It counts the partitioning's blocks that contain exactly k elements, written $\sum_i [|B_i| = k]$. It is exchangable with respect to the elements $[n]$, but its weighted average over a sample set is difficult to interpret. Let us illustrate the problem by a practical example, to which we will return in the formulations:

$$E_3 \ = \ \{Z^{(1)}, Z^{(2)}, Z^{(3)}\} \qquad \begin{array}{ll} Z^{(1)} \ = \{\{1,3,6,7\},\{2\},\{4,5\}\} & S_1 \ = \{1,2,3,4\} \\ Z^{(2)} \ = \{\{1,3,6\},\{2,7\},\{4,5\}\} & S_2 \ = \{1,3,6,7\} \\ Z^{(3)} \ = \{\{1,2,3,6,7\},\{4,5\}\} & S_3 \ = \{1,2,3\} \end{array} \tag{5.4}$$

Suppose that E_3 represents interactions among seven genes. We want to compare the subsets of these genes S_1, S_2, S_3. The *projection* of a partitioning $Z \vdash [n]$ onto $S \subset [n]$ is defined as the set of non-empty intersections between S and $B \in Z$. Projection onto

S induces a partitioning of S.

$$PROJ(Z,S) \;=\; \{B \cap S\}_{B\in Z}\backslash\{\emptyset\} \qquad \Rightarrow \qquad PROJ(Z,S) \vdash S \tag{5.5}$$

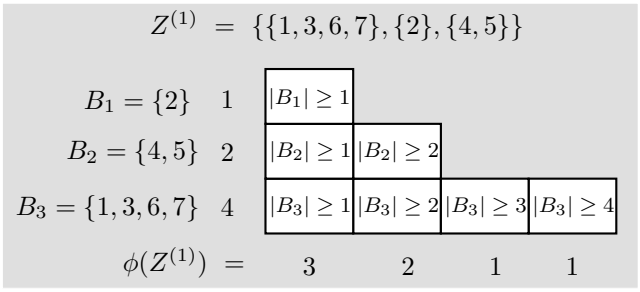
Let us represent gene interactions in $Z^{(1)}$ and $Z^{(2)}$ by projecting them onto each subset:

$$\begin{aligned}
PROJ(Z^{(1)},S_1) &= \{\{1,3\},\{2\},\{4\}\} & PROJ(Z^{(2)},S_1) &= \{\{1,3\},\{2\},\{4\}\} \\
PROJ(Z^{(1)},S_2) &= \{\{1,3,6,7\}\} & PROJ(Z^{(2)},S_2) &= \{\{1,3,6\},\{7\}\} \\
PROJ(Z^{(1)},S_3) &= \{\{1,3\},\{2\}\} & PROJ(Z^{(2)},S_3) &= \{\{1,3\},\{2\}\}
\end{aligned} \tag{5.6}$$

Comparing S_1 to S_2, we can say that S_1 is 'more segmented' than S_2, and therefore genes in S_2 should be more closely related than those in S_1. However, it is more subtle and difficult to compare S_2 to S_3. A clear understanding would allow us to explore the subsets $S \subset [n]$ in an informed manner. In the following section, we develop a novel and general approach based on block sizes that opens up a systematic method for analyzing sample sets over partitionings and feature allocations.

5.3. Cumulative statistics to represent structure

We define *cumulative block size distribution*, or 'cumulative statistic' in short, as the function $\phi_k(Z) = \sum_i [|B_i| \geq k]$, which counts the partitioning's blocks of size at least k. We can rewrite the previous statistics: number of blocks as $\phi_1(Z)$, exact block size distribution as $\phi_k(Z) - \phi_{k+1}(Z)$, and pairwise occurence as $\phi_2(PROJ(Z,\{a,b\}))$. Moreover, cumulative statistics satisfy the following property: for partitionings of $[n]$, $\phi(Z)$ always sums up to n, just like a probability mass function that sums up to 1. When blocks of Z are sorted according to their sizes and the indicators $[|B_i| \geq k]$ are arranged on a matrix as in Figure 5.1a, they form a Young diagram, showing that $\phi(Z)$ is always the conjugate partition of the integer partition of Z. As a result, $\phi(Z)$ as well as weighted averages over several $\phi(Z)$ always sum up to n, just like taking averages over probability mass functions (Figure 5.2). Therefore, cumulative statistics



(a) Cumulative statistic for a partitioning (a) For its projection onto a subset

Figure 5.1. Young diagrams show the conjugacy of the cumulative statistic

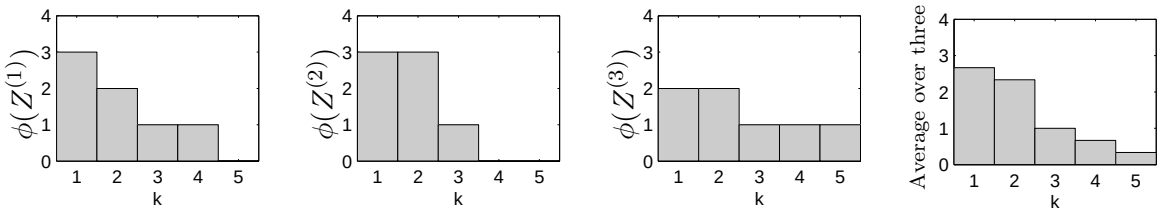
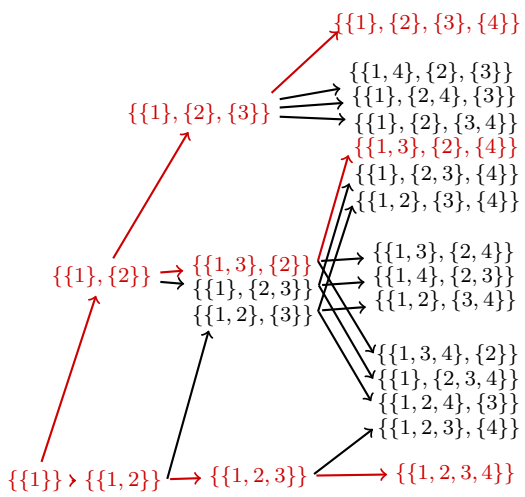


Figure 5.2. Cumulative statistics of 3 examples and their average, each sum up to 7.

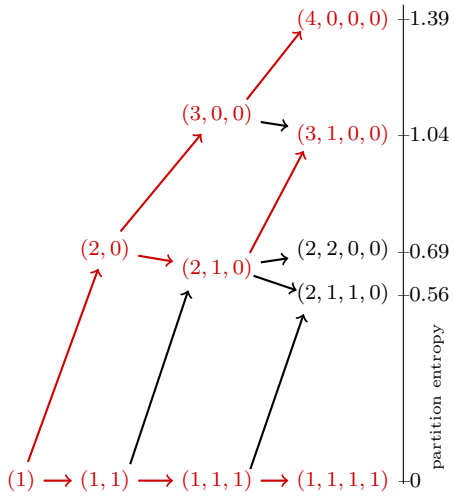
of a random partitioning 'conserve mass'. In the case of feature allocations, since elements can be omitted or repeated, this property does not hold.

$$Z \vdash [n] \quad \Rightarrow \quad \sum_{k=1}^{n} \phi_k(Z) = n \quad \Rightarrow \quad \sum_{k=1}^{n} \langle\ \phi_k(Z)\ \rangle_{\pi(Z)} = n \tag{5.7}$$

When we project the partitioning Z onto a subset $S \subset [n]$, the resulting vector $\phi(PROJ(Z,S))$ will then sum up to $|S|$ (Figure 5.1b). A 'taller' Young diagram implies a 'more segmented' subset. We can form a partitioning Z by inserting elements $1, 2, 3, 4, \ldots$ into its blocks (Figure 5.3a). In such a scheme, each step brings a new element and requires a new decision that will depend on all previous decisions. It would be better if we could determine the whole path by few initial decisions. Now suppose that we know Z from the start and we generate an incremental sequence of subsets $S_1 = \{1\}$, $S_2 = \{1,2\}$, $S_3 = \{1,2,3\}$, $S_4 = \{1,2,3,4\}$, ... according to a permutation of $[n]$: $\sigma = (1, 2, 3, 4, \ldots)$. We can then represent any path in Figure 5.3a by a sequence of $PROJ(Z, S_i)$ and determine the whole path by two initial parameters: Z and σ. The resulting tree can be simplified by representing the partitionings by their cumulative statistics instead of their blocks (Figure 5.3b).



(a) Form partitioning by inserting elements



(b) Form the vector by inserting elements

1						
1	1					
1	1	1				
1	1	1	1			
1	1	1	1	1		
1	1	1	1	1	1	
1	1	1	1	1	1	1

(c) All in one block

1	1						
2	2	0					
3	2	1	0				
4	3	1	0	0			
5	3	2	0	0	0		
6	3	2	1	0	0	0	
7	3	2	1	1	0	0	0

(d) COD matrix $\Delta(Z^{(1)}, (1, \ldots, 7))$

1	1						
2	2	0					
3	3	0	0				
4	4	0	0	0			
5	5	0	0	0	0		
6	6	0	0	0	0	0	
7	7	0	0	0	0	0	0

(e) Each to a new block

Figure 5.3. COD matrices correspond to the red dotted paths on the trees above

Based on this concept, we define *cumulative occurence distribution* (COD) as the triangular matrix of incremental cumulative statistic vectors, written $\Delta_{i,k}(Z, \sigma) = \phi_k(PROJ(Z, S_i))$ where $Z \vdash [n]$, σ is a permutation of $[n]$ and $S_i = \{\sigma_1, \ldots, \sigma_i\}$ for $i \in \{1, \ldots, n\}$. COD matrices for two extreme paths (Figure 5.3c, 5.3e) and for the example partitioning $Z^{(1)}$ (Figure 5.3d) are shown. For partitionings, ith row of a COD matrix always sums up to i, even when averaged over a sample set as in Figure 5.6.

$$Z \vdash [n] \quad \Rightarrow \quad \sum_{k=1}^{i} \Delta_{i,k}(Z, \sigma) = i \quad \Rightarrow \quad \sum_{k=1}^{i} \langle\, \Delta_{i,k}(Z, \sigma) \,\rangle_{\pi(Z)} = i \qquad (5.8)$$

Expected COD matrix of a random partitioning expresses (1) cumulation of elements by the differences between its rows, and (2) cumulation of block sizes by the differences between its columns. As an illustrative example, consider $\pi(Z) = CRP(Z|\alpha, d)$. Since CRP is exchangable and projective, its expected cumulative statistic $\langle \phi(Z) \rangle_{\pi(Z)}$ for

n elements depends only on its hyperparameters (α, d). As a result, its expected COD matrix $\Delta = \langle \Delta(Z, \sigma) \rangle_{\pi(Z)}$ is independent of σ, and it satisfies an incremental formulation with the parameters (α, d) over the indices $i \in \mathbb{N},\ k \in \mathbb{Z}^+$:

$$\Delta_{0,k} = 0 \qquad \Delta_{i+1,k} = \Delta_{i,k} + \begin{cases} \frac{\alpha + d\Delta_{i,k}}{i+\alpha} & \text{if } k = 1 \\ \frac{(k-1-d)(\Delta_{i,k-1} - \Delta_{i,k})}{i+\alpha} & \text{otherwise} \end{cases} \tag{5.9}$$

By allowing $k = 0$ and setting $\Delta_{i,0} = -\frac{\alpha}{d}$, and $\Delta_{0,k} = 0$ for $k > 0$ as the two boundary conditions, the same matrix can be formulated by a difference equation over the indices $i \in \mathbb{N},\ k \in \mathbb{N}$:

$$(\Delta_{i+1,k} - \Delta_{i,k})(i + \alpha) = (\Delta_{i,k-1} - \Delta_{i,k})(k - 1 - d) \tag{5.10}$$

By setting $\Delta = \Delta^{(0)}$ we get an infinite sequence of matrices $\Delta^{(m)}$ that satisfy the same equation:

$$(\Delta^{(m)}_{i+1,k} - \Delta^{(m)}_{i,k})(i + \alpha) = (\Delta^{(m)}_{i,k-1} - \Delta^{(m)}_{i,k})(k - 1 - d) = \Delta^{(m+1)}_{i,k} \tag{5.11}$$

Therefore, expected COD matrix of a CRP-distributed random partitioning is at a constant 'equilibrium' determined by α and d. This example shows that the COD matrix can reveal specific information about a distribution over partitionings; of course in practice we encounter non-exchangeable and almost arbitrary distributions over partitionings (e.g., the posterior distribution of an infinite mixture), therefore in the following section we will develop a measure to quantify this information.

5.4. Entropy to quantify segmentation

Shannon's entropy [41] can be an appropriate quantity to measure 'segmentation' with respect to partitionings, which can be interpreted as probability distributions [43, 44]. Since this interpretation does not cover feature allocations, we will make an alternative, element-based definition of entropy.

How does a block B inform us about its elements? Each element has a proportion $1/|B|$, let us call this quantity *per-element segment size*. Information is zero for $|B| = n$, since $1/n$ is the minimum possible segment size. If $|B| < n$, the block supplies positive information since the segment size is larger than minimum, and we know that *its segment size could be smaller if the block were larger*. To quantify this information, we define *per-element information* for a block B as the integral of segment size $1/s$ over the range $[|B|, n]$ of block sizes that make this segment smaller (Figure 5.4).

$$\mathrm{pei}_n(B) \;=\; \int_{|B|}^{n} \frac{1}{s}\, ds \;=\; \log \frac{n}{|B|} \tag{5.12}$$

In $\mathrm{pei}_n(B)$, n is a 'base' that determines the minimum possible per-element segment size. Since segment size expresses the *significance* of elements, the function integrates segment sizes over the block sizes that make the elements *less significant*. This definition is comparable to the well-known *p-value*, which integrates probabilities over the values that make the observations *more significant*. We can then compute the per-element information supplied by a partitioning Z, by taking a weighted average over its blocks, since each block $B \in Z$ supplies information for a different proportion $|B|/n$ of the elements being partitioned. For large n, weighted per-element information

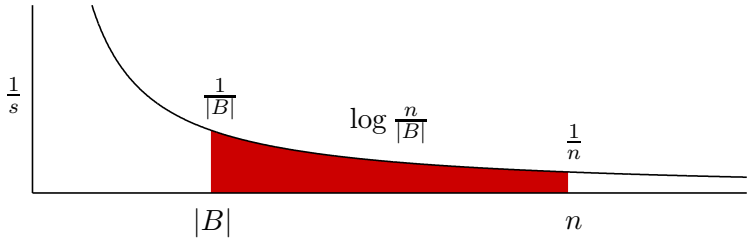


Figure 5.4. Per-element information

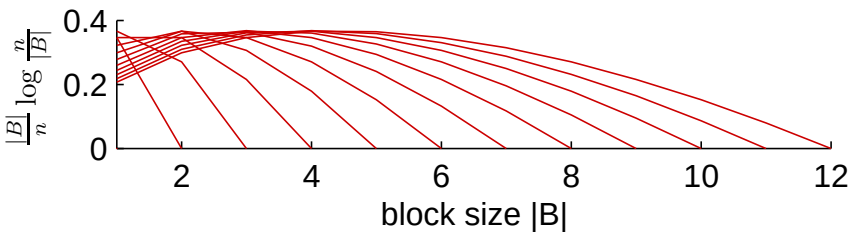


Figure 5.5. Weighted information

reaches its maximum near $|B| \approx n/2$ (Figure 5.5). Total weighted information for Z gives Shannon's entropy function [41] which can be written in terms of the cumulative statistics (assuming $\phi_{n+1} = 0$):

$$H(Z) \;=\; \sum_{i=1}^{|Z|} \frac{|B_i|}{n}\, \mathrm{pei}_n(B_i) \;=\; \sum_{i=1}^{|Z|} \frac{|B_i|}{n} \log \frac{n}{|B_i|} \;=\; \sum_{k=1}^{n} (\phi_k(Z) - \phi_{k+1}(Z)) \frac{k}{n} \log \frac{n}{k} \tag{5.13}$$

Entropy of a partitioning increases as its elements become more segmented among themselves. A partitioning with a single block has zero entropy, and a partitioning with n blocks has the maximum entropy $\log n$. Nodes of the tree we examined in the previous section (Figure 5.3b) were vertically arranged according to their entropies. On the extended tree (Figure 5.7), nth column of nodes represent the possible partitionings of n. This tree serves as a 'grid' for both $H(Z)$ and $\phi(Z)$, as they are linearly related with the general coefficient $(\frac{k}{n} \log \frac{n}{k} - \frac{k-1}{n} \log \frac{n}{k-1})$. A similar grid for feature allocations can be generated by inserting nodes for cumulative statistics that do not conserve mass. We compute *projection entropy* $H(PROJ(Z, S))$ to quantify the segmentation of a subset S. To understand this function, we compare it to *subset occurence* in

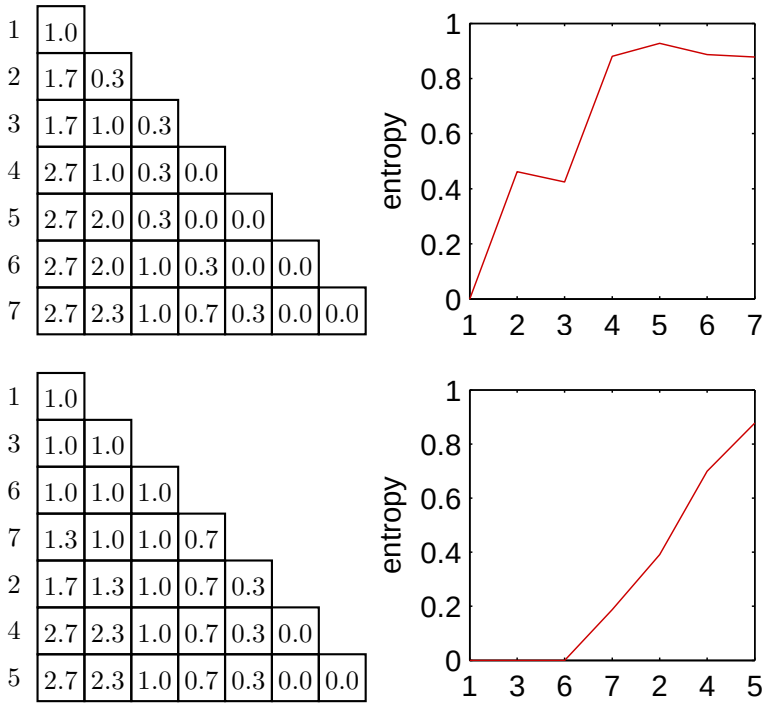


Figure 5.6. CODs and entropies over E_3 for $(1, 2, 3, 4, 5, 6, 7)$ and $(1, 3, 6, 7, 2, 4, 5)$

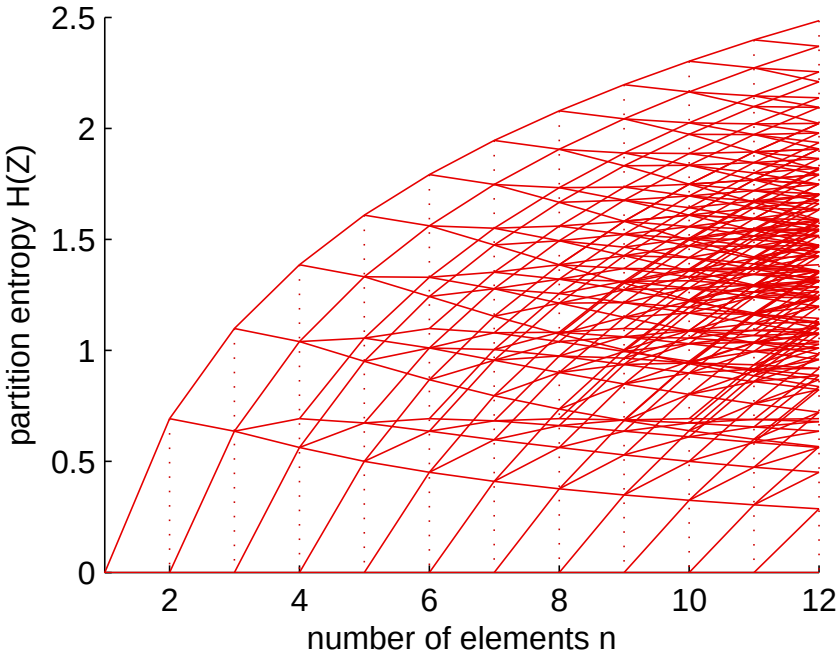


Figure 5.7. $H(Z)$ in constructing Z

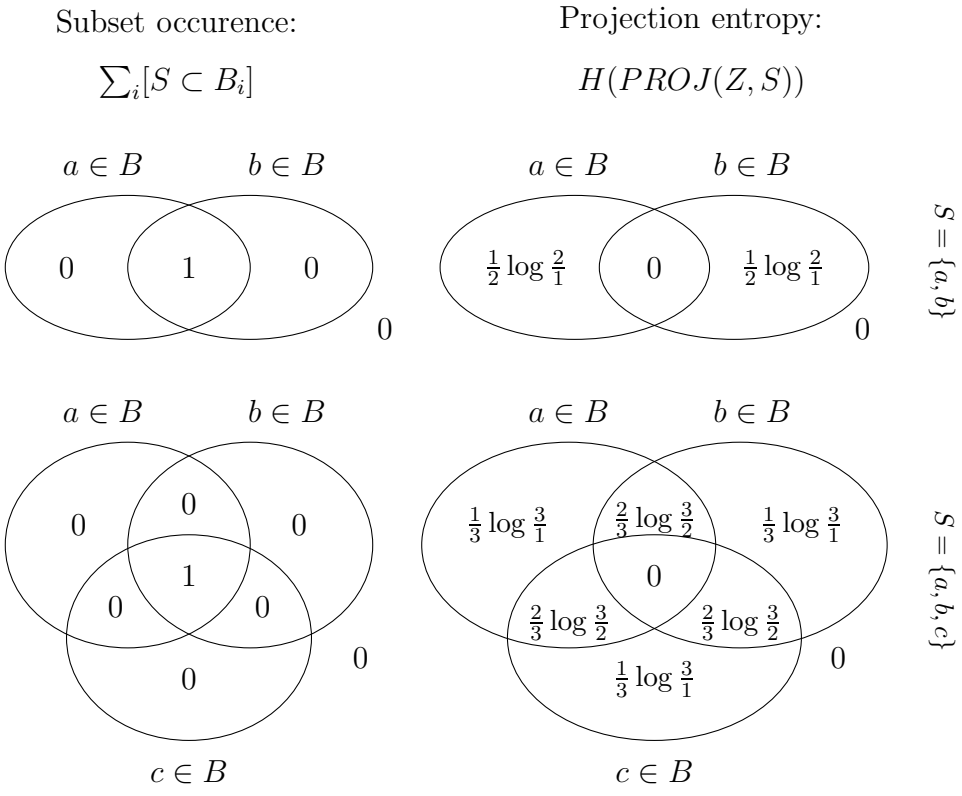


Figure 5.8. Comparing two statistics

Figure 5.8. Subset occurence acts as a 'score' that counts the 'successful' blocks that contain all of S, whereas projection entropy acts as a 'penalty' that quantifies how much S is being divided and segmented by the given blocks $B \in Z$. A partitioning Z and a permutation σ of its elements induce an *entropy sequence* $(h_1, \ldots, h_n)$ such that $h_i(Z, \sigma) = H(PROJ(Z, S_i))$ where $S_i = \{\sigma_1, \ldots, \sigma_i\}$ for $i \in \{1, \ldots, n\}$. To find subsets of elements that are more closely related, one can seek permutations σ that keep the entropies low. The generated subsets S_i will be those that are less segmented by $B \in Z$. For the example problem, the permutation $1, 3, 6, 7, \ldots$ keeps the expected entropies lower, compared to $1, 2, 3, 4, \ldots$ (Figure 5.6).

5.5. Entropy agglomeration and experimental results

We want to summarize a sample set using the proposed statistics. Permutations that yield lower entropy sequences can be meaningful, but a feasible algorithm can only involve a small subset of the $n!$ permutations. We define *entropy agglomeration* (EA) algorithm, which begins from 1-element subsets, and merges in each iteration the pair of subsets that yield the minimum expected entropy:

(i) Initialize $\Psi \leftarrow \{\{1\}, \{2\}, \ldots, \{n\}\}$.
(ii) Find the subset pair $\{S_a, S_b\} \subset \Psi$ that minimizes the entropy $\langle\, H(PROJ(Z, S_a \cup S_b))\, \rangle_{\pi(Z)}$.
(iii) Update $\Psi \leftarrow (\Psi \backslash \{S_a, S_b\}) \cup \{S_a \cup S_b\}$.
(iv) If $|\Psi| > 1$ then go to 2.
(v) Generate the dendrogram of chosen pairs by plotting minimum entropies for every split.

The resulting dendrogram for the example partitionings are shown in Figure 5.9a. The subsets $\{4, 5\}$ and $\{1, 3, 6\}$ are shown in individual nodes, because their entropies are zero. Besides using this dendrogram as a general summary, one can also generate more specific dendrograms by choosing specific elements or specific parts of the data. For a detailed element-wise analysis, entropy sequences of particular permutations σ can be assessed. Entropy Agglomeration is inspired by 'agglomerative clustering', a standard

approach in bioinformatics [20]. To summarize partitionings of gene expressions, [23] applied agglomerative clustering by pairwise occurences. Although very useful and informative, such methods remain 'heuristic' because they require a 'linkage criterion' in merging subsets. EA avoids this drawback, since projection entropy is already defined over subsets.

To test the proposed algorithm, we apply it to partitionings sampled from infinite mixture posteriors. In the first three experiments, data is modeled by an infinite mixture of Gaussians, where $\alpha = 0.05, d = 0$, $p(\theta) = \mathcal{N}(\theta|0, 5)$ and $F(x|\theta) = \mathcal{N}(x|\theta, 0.15)$ (see Equation 5.1). Samples from the posterior are used to plot the histogram over the number of blocks, pairwise occurences, and the EA dendrogram. Pairwise occurences are ordered according to the EA dendrogram. In the fourth experiment, EA is directly applied on the data. We describe each experiment and make observations:

1) Synthetic data (Figure 5.9b): 30 points on $\mathbb{R}^2$ are arranged in three clusters. Plots are based on 450 partitionings from the posterior. Clearly separating the three clusters, EA also reflects their qualitative differences. The dispersedness of the first cluster is represented by distinguishing 'inner' elements 1, 10, from 'outer' elements 6, 7. This is also seen as shades of gray in pairwise occurences.

2) Iris flower data (Figure 5.9c): This well-known dataset contains 150 points on $\mathbb{R}^4$ from three flower species [45]. Plots are based on 150 partitionings obtained from the posterior. For convenience, small subtrees are shown as single leaves and elements are labeled by their species. All of 50 A points appear in a single leaf, as they are clearly separated from B and C. The dendrogram automatically scales to cover the points that are more uncertain with respect to the distribution.

3) Galactose data (Figure 5.9d): This is a dataset of gene expressions by 820 genes in 20 experimental conditions [46]. First 204 genes are chosen, and first two letters of gene names are used for labels. Plots are based on 250 partitionings from the posterior. 70 RP (ribosomal protein) genes and 12 HX (hexose transport) genes appear in individual leaves. In the large subtree on the top, an 'outer' grouping of 19

genes (circles in data plot) is distinguished from the 'inner' long tail of 68 genes.

4) IGO (Figure 5.9e): This is a dataset of intergovernmental organizations (IGO) [47] that contains IGO memberships of 214 countries through the years 1815-2000. In this experiment, we take a different approach and apply EA directly on the dataset interpreted as a sample set of single-block feature allocations, where the blocks are IGO-year tuples and elements are the countries. We take the subset of 138 countries that appear in at least 1000 of the 12856 blocks. With some exceptions, the countries display a general ordering of continents. From the 'outermost' continent to the 'innermost' continent they are: Europe, America-Australia-NZ, Asia, Africa and Middle East.

5.6. Discussion

In this section, we developed a novel approach for summarizing sample sets of partitionings and feature allocations. After presenting the problem, we introduced cumulative statistics and cumulative occurence distribution matrices for each of its permutations, to represent a sample set in a systematic manner. We defined per-element information to compute entropy sequences for these permutations. We developed *entropy agglomeration* (EA) algorithm that chooses and visualises a small subset of these entropy sequences. Finally, we experimented with various datasets to demonstrate the method.

Entropy agglomeration is a simple algorithm that does not require much knowledge to implement, but it is conceptually based on the cumulative statistics we have presented. Since we primarily aimed to formulate a useful algorithm, we only made the essential definitions, and several points remain to be elucidated. For instance, cumulative statistics can be investigated with respect to various nonparametric priors. Our definition of per-element information can be developed with respect to information theory and hypothesis testing. Last but not least, algorithms like entropy agglomeration can be designed for summarization tasks concerning various types of combinatorial sample sets.

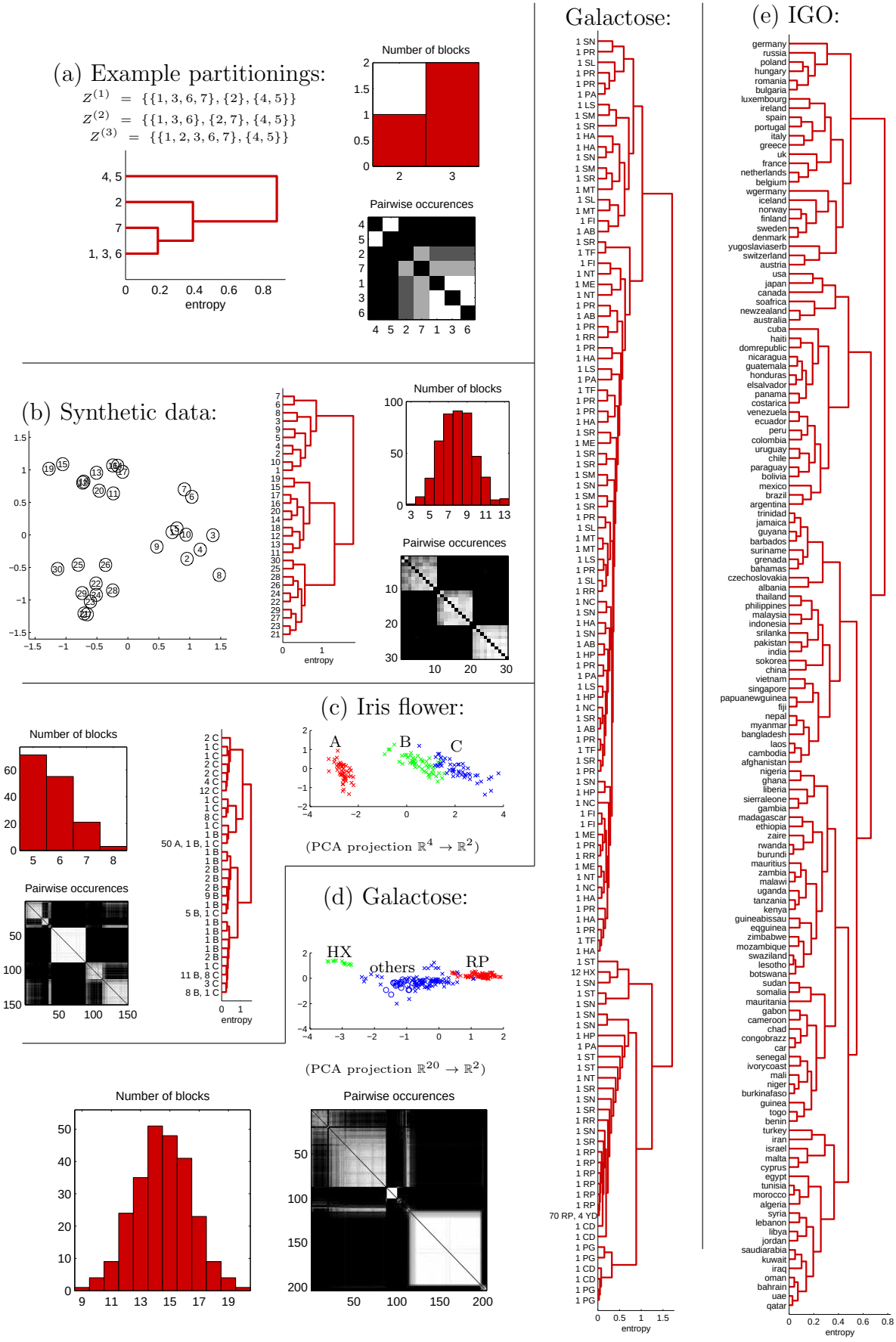
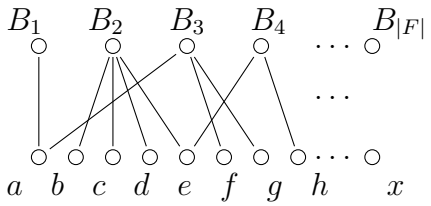


Figure 5.9. Entropy agglomeration and results from experiments (See text)

6. CLUSTERING WORDS BY PROJECTION ENTROPY

In this chapter we present the application of entropy agglomeration (EA) to cluster the words of a literary text. EA is a greedy agglomerative procedure that minimizes projection entropy (PE), a function that can quantify the segmentedness of an element set. To apply it, the text is reduced to a feature allocation, a combinatorial object to represent the word occurences in the text's paragraphs. The experiment results demonstrate that EA, despite its reduction and simplicity, is useful in capturing significant relationships among the words in the text. This procedure was implemented in Python and published as a free software: REBUS.

Problems in natural language processing involve many difficulties due to the variety of subtleties and ambiguities in languages. On the other hand, these problems can be familiar to people in different fields, since the spoken and written languages constitute our common ground. It is especially favorable to approach these difficulties with generic statistical concepts, since these can also be utilized in other challenging fields such as bioinformatics. Here we demonstrate the use of entropy agglomeration (EA), a recently introduced algorithm [28], by clustering the words of a literary text. By following the state-of-the-art NLP methods, we assume that *words* are the basic elements of a text [48] [3]. Moreover, to approach text analysis in a simpler form, we disregard all sequential ordering, considering each paragraph as a subset of words, and the whole text as a set of paragraphs. A statistical analysis of such data would conventionally be formulated in terms of joint probabilities of word sets to *co-occur* in the paragraphs. Such a probabilistic formulation can be extended to potentially infinite number of words by using Bayesian nonparametric models [49]. However, we take a different approach: we compute projection entropies (PE) of word sets, and use them in a clustering algorithm called entropy agglomeration (EA) to find the meaningful correlations among the words in the text. And we will briefly review the statistical concepts that were introduced in [28]. In the following sections, we define how the input data is represented by feature allocations, quantified by projection entropies and clustered by entropy agglomeration. We describe the experiment procedure on the



Entropy: $H(F) = \sum_{i=1}^{|F|} \frac{|B_i|}{n} \log \frac{n}{|B_i|}$

Projection: $PROJ(F, S) = \{B \cap S\}_{B \in F} \backslash \{\emptyset\}$

Projection entropy: $H(PROJ(F, S))$

Figure 6.1. Text is represented by a feature allocation.

way, and finally present the results, which demonstrate that our algorithm is able to capture a variety of meaningful relationships among the words of the input text. An additional discussion of projection entropy in comparison to co-occurence is included in the fourth section.

6.1. The input text and its representation

We picked *Ulysses* (1922) by James Joyce to be the input text. It consists of 7437 paragraphs and contains 29327 distinct words. But we only want to know which words occur in which paragraphs. This information is illustrated as a bipartite graph in Figure 6.1 where word elements below are linked to the paragraphs above them. Any text with n distinct words will be represented by a feature allocation defined as follows: A feature allocation of a set of elements $[n] = \{1, 2, \ldots, n\}$, is a multiset of blocks $F = \{B_1, \ldots, B_{|F|}\}$ such that $B_i \subset [n]$ and $B_i \neq \emptyset$ for all $i \in \{1, \ldots, n\}$. The blocks $B_1, \ldots, B_{|F|}$ in this definition will represent the paragraphs of the input text.

See the definitions given in Figure 6.1. Entropy quantifies the segmentedness of elements with respect to the blocks of F. It becomes maximum at the blocks that include about half of the elements. Projection of F onto S restricts the scope of F to this subset, functioning like a filter to focus on this particular subset. Projection entropy computes the segmentedness for a particular subset, by projecting F onto it (See [28]). If a subset has low PE, we say its elements have *entropic correlation*. The projection size $|PROJ(F, \{a\})|$ of an element a is the number of blocks that include it. In our case, a indicates a word and its projection size is the number of paragraphs it occurs in.

6.2. Clustering the word sets

Ulysses is reduced, but the feature allocation still contains too many words. Word sets of manageable sizes are needed for analysis. Nine word sets are assembled by restricting their projection sizes (number of paragraphs they occur in) in ranges 10, 11, 12-13, 15-17, 20-25, 30-39, 40-59, 60-149 and 150-7020. Then, the full feature allocation is projected onto each of these word sets, and EA is run on each of these projected feature allocations. Here is the pseudocode for the EA algorithm:

(i) Initialize $\Psi \leftarrow \{\{1\}, \{2\}, \ldots, \{n\}\}$.
(ii) Find the subset pair $\{S_a, S_b\} \subset \Psi$ that minimizes the entropy $H(PROJ(F, S_a \cup S_b))$.
(iii) Update $\Psi \leftarrow (\Psi \backslash \{S_a, S_b\}) \cup \{S_a \cup S_b\}$.
(iv) If $|\Psi| > 1$ then go to 2.
(v) Generate the dendrogram of chosen pairs by plotting minimum entropies for every bifurcation.

EA generates a dendrogram for each word set to show the entropic correlations among its elements. Dendrograms are diagrams commonly used to display results of hierarchical clustering algorithms [20, 28]. Sample word pairs from EA dendrograms are shown in Table 6.1 to illustrate the variety of entropic correlations detected by the algorithm. These correlations indicate a diversity of semantic relationships: black-white, south-north are antonyms; then-now, former-latter, came-went are contraries; female-male, Eve-Adam, you-I indicate reciprocities; red-green are colors; four-five, nine-eleven are quantities; his-he, her-she, me-my, us-our, them-their, thy-thou are inflections of different pronouns; hear-heard, looking-looked, smile-smiled, pouring-poured are inflections of different verbs; ireland-irish is the inflection of a nation. Some of the other contextual correlations of expressions, things and figures are also enumerated. Entropic correlations cover an interesting range of meanings.

Differences:	**Inflections:**	**Expressions:**		**Figures:**
black – white	his – he	ah – sure	window – seen	girl – sweet
south – north	her – she	ay – eh	hand – eyes	dame – joy
then – now	me – my	darling – perfume	face – head	females – period
former – latter	us – our	thank – please	cup – tea	wife – world
came – went	them – their	**Things:**	plate – fork	woman – behind
red – green	thy – thou	ocean – level	food – eating	gentleman – friend
female – male	hear – heard	waves – waters	job – business	gentlemen – friends
eve – adam	looking – looked	river – boat	sell – trade	priest – quietly
you – I	smile – smiled	moon – stars	slice – quantity	reverend – blessed
four – five	pouring – pour	birds – fly	family – memory	christ – jew
nine – eleven	ireland – irish	grass – fields	road – city	human – live
			system – distance	

Table 6.1. Sample word pairs to illustrate the entropic correlations captured by EA.

6.3. On the meaning of projection entropy

Projection entropy (PE) is a useful guiding principle in exploring significant element-wise relationships in combinatorial data. But it has a meaning that is quite different from conventional statistical methods. Therefore in this section, we would like to discuss the meaning of projection entropy in comparison to a well-known quantity, *co-occurence of elements*, which is used for similar purposes in probabilistic modeling. The actualizations of these quantities' values for 2 and 3 elements are illustrated in Figure 5.8. Firstly, as pointed out in [28], contrary to the positive sense of co-occurence as scoring the blocks where all elements co-occur; PE has a negative sense of penalizing the blocks that divide and separate them. Secondly, co-occurence is non-zero only at the blocks that include all of the elements, leaving out the blocks that exclude any of them. But PE leaves out the blocks that include all elements as well as the blocks that exclude all elements; it is non-zero only at partial inclusions: at the blocks that include some elements while excluding other elements. This makes PE a flexible quantity that can adjust to the blocks where any part of the elements overlap, whereas co-occurence is a rigid quantity that can only adjust to the blocks where all of the elements overlap.

Assume that we have a cluster S whose elements have constant projection sizes. If these elements do not overlap at any of the blocks that include any of them, the cluster's PE will take the maximum value: the sum of projection sizes multiplied by $\frac{\log|S|}{|S|}$. If these elements overlap at all of the blocks that include them, PE will be zero by definition. Moreover, any additional overlapping among the elements will decrease the cluster's PE, if the projection sizes are kept constant. Therefore, lower PE indicates higher overlapping in the cluster, relative to its elements' projection sizes. To express this element-wise overlapping indicated by a lower PE, we say that these elements have an *entropic correlation* at the blocks that include them.

To understand how PE functions in entropy agglomeration, let us examine its effect for a word pair. Assume that we have a pair from the set of words that occur exactly in 10 paragraphs. We know that (1) projection sizes for both words are 10, (2) co-occurence would count the blocks that include both of them, (3) PE would count the blocks that include one of them. For this particular case, these two quantities are directly proportional: an increase in co-occurence by 1 would decrease PE by $\log 2$. This makes them practically equivalent. However, if there are several projection sizes like 20-25, words with larger projections can have partial occurences more frequently; co-occurence would ignore these occurences, but PE may count them to penalize the elements for occuring 'unnecessarily'.

In conclusion, we developed in this work a text analysis tool that visualizes the entropic correlations among the words of a given text by applying entropy agglomeration to its paragraphs, and published this procedure as a free software: REBUS [50]. We demonstrated the utility of this procedure by clustering the words of a literary text using this tool.

7. CLUSTERnGO (CnG) MODELING PLATFORM

In this chapter, we describe the processes of the four-phase pipeline of our bioinformatics application CLUSTERnGO (CnG) based on the IMPLS model.

Simple bioinformatic tools are frequently used to analyse time-series datasets regardless of their ability to deal with transient phenomena, limiting the meaningful information that may be extracted from them. This situation requires the development and exploitation of tailor-made, easy-to-use, and flexible tools designed specifically for the analysis of time-series datasets.

We present a novel statistical application called CLUSTERnGO, which uses a model-based clustering algorithm that fulfils this need. This algorithm involves two components of operation. Component 1 constructs a Bayesian non-parametric model (Infinite Mixture of Piecewise Linear Sequences) and Component 2, which applies a novel clustering methodology (Two-Stage Clustering). The software can also assign biological meaning to the identified clusters using an appropriate ontology. It applies multiple hypothesis testing to report the significance of these enrichments. The algorithm has a four-phase pipeline. The application can be executed using either command-line tools or a user-friendly Graphical User Interface. The latter has been developed to address the needs of both specialist and non-specialist users. We use three diverse test cases to demonstrate the flexibility of the proposed strategy. In all cases, CLUSTERnGO not only outperformed existing algorithms in assigning unique GO term enrichments to the identified clusters, but also revealed novel insights regarding the biological systems examined, which were not uncovered in the original publications.

The C++ and QT source codes, the GUI applications for Windows, OS X and Linux operating systems and user manual are freely available for download under the GNU GPL v3 license [51].

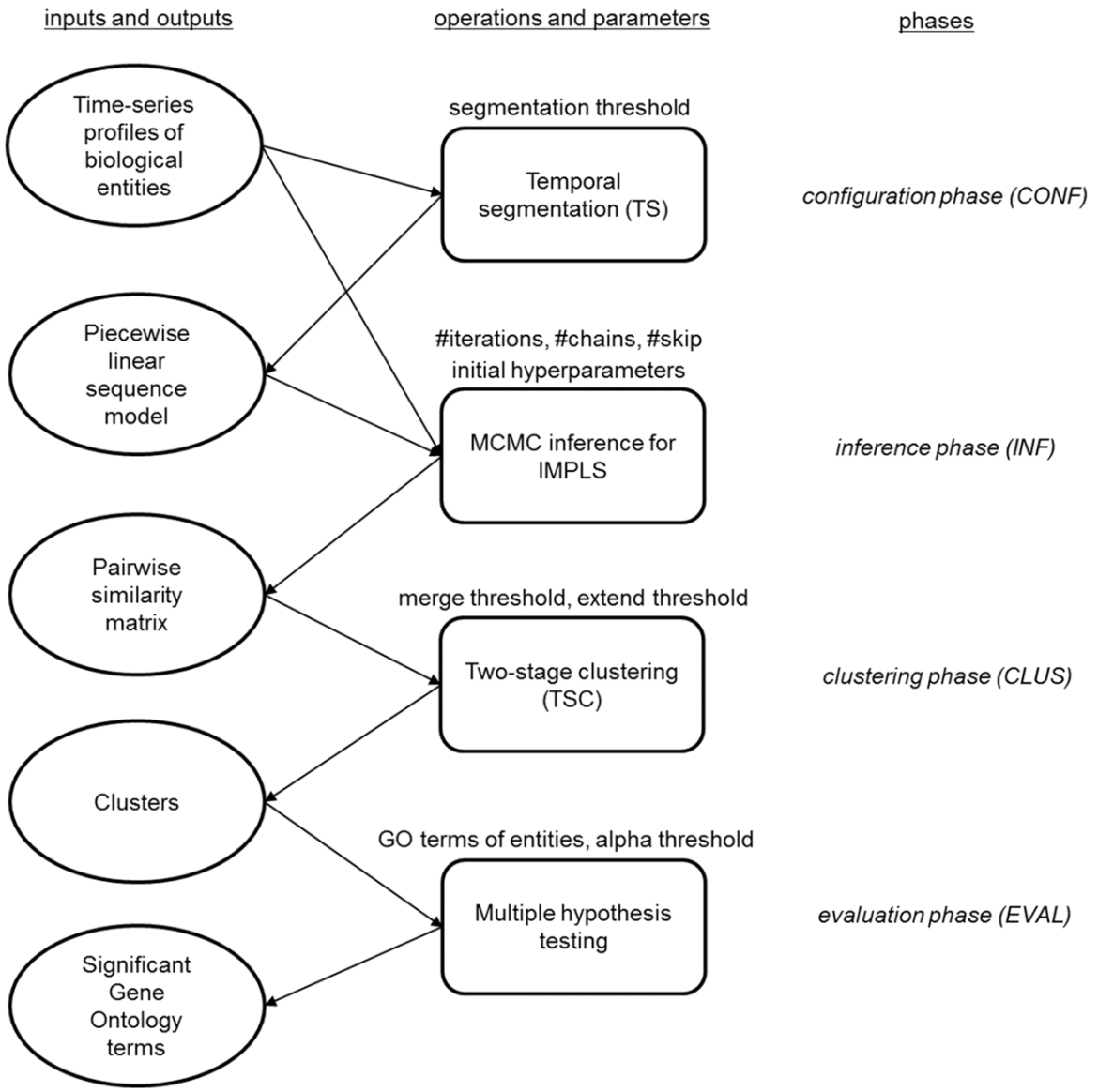


Figure 7.1. Structural design of the algorithm. Operations in each of the four phases are shown in rectangular boxes, with any parameters (if applicable) written above them. Inputs and outputs of these operations are indicated in oval boxes.

7.1. Algorithm

The algorithm we propose involves a single process of clustering analysis and consists of four successive phases: configuration (CONF), inference (INF), clustering (CLUS), and evaluation (EVAL) (Figure 7.1). Inputs and outputs of these operations follow successive steps in a single pipeline. The process, taken as a whole, receives an input dataset of dynamic profiles and assigns the profiles into an optimal number of

clusters based on the model determined by the user as well as reporting an output of statistically significant Gene Ontology (GO) terms that characterize those clusters of entities, whenever applicable. In this section, we describe the functioning of each of the four phases in the CnG algorithm pipeline.

7.1.1. Configuration Phase (CONF)

The most important feature of datasets on transitions is the dependence of the value of each variable on its value at the preceding time point. Therefore, it is important to account for this information during the identification of clusters of entities displaying similar behaviour over time. Our approach involves building a model based on the experimental input as well as the initial design of the experiment to account for the dependencies between consecutive time points in dealing with transient phenomena. CONF is the phase in our algorithm that configures this model.

Our algorithm models the given time-series dataset by an infinite mixture of piecewise linear sequences (IMPLS). IMPLS is a special infinite mixture model whose mixture components are distributed around piecewise linear sequences (PLS). PLS assumes a particular segmentation of time points, where in each segment corresponding to a given time period, the measured level of the clustered entities is assumed to linearly increase, decrease, or constitutively stay constant. PLS model is illustrated in Figure 7.2.

CONF is the initial phase for configuring the probabilistic model for Bayesian inference. It can be configured manually by specifying a custom segmentation of time points for the PLS model, or it can be configured semi-automatically. In the semi-automatic mode, it takes the time-course profiles of the biological entities in the dataset as its input and, by applying temporal segmentation (TS) to its time points, produces the piecewise linear sequence (PLS) model that will be used in the next phase. TS has a single parameter: the segmentation threshold. TS determines which time samples show similar behaviour by taking values for each of the time points over the whole dataset, and running a standard average-linkage hierarchical agglomerative clustering

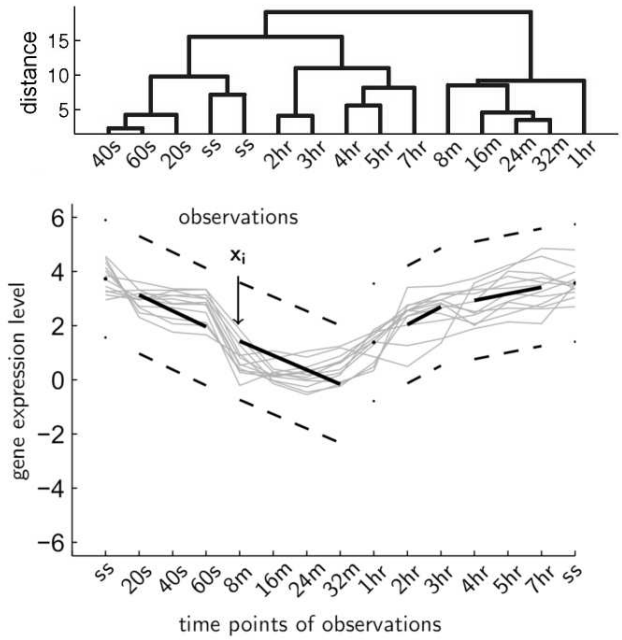


Figure 7.2. A dendrogram generated in CONF shows correlation in segments. Times-series observations xi are assumed to be distributed around Piecewise Linear Sequences.

procedure based on their pairwise Gaussian distances. By applying a threshold on the resulting dendrogram at a certain value, which we call the segmentation threshold, time samples can be grouped such that they make up a piecewise linear sequence. The threshold is determined by the end-user in order to represent the sub-sequences of time points that are known to have a linear succession in the experimental setup, and by the temporal segments in PLS model. It is possible to trace how the groupings change as the threshold is varied, thus allowing the user to adjust the time segments until the most biologically meaningful segmentation, based on the experimental design, is obtained. The constructed PLS models are then used to determine the probabilistic model in the inference phase. Although one can also take PLS segmentation as a probabilistic variable to be inferred, in CnG, we choose to keep it as a user-defined model parameter.

Biological experiments are usually designed to seek answers to specific questions and have an a priori hypothesis to be tested. This hypothesis is taken into account in

the design of an experiment to determine the type and duration of the perturbations as well as the sample collection regime. In CONF, the users can construct their own models that integratively, take into account both the design of experiment and the data collected from those experiments. Naturally the a priori expectations arising from the initial design of the experiment may not always meet the actual outcome represented by the data generated. This step may assume the role of an integral check point highlighting important intrinsic characteristics of the data. It may: (i) capture novel behaviour emerging from the data that was initially unexpected when designing the experiment, or (ii) highlight inconsistencies or inaccuracies within the data caused either by the experiment itself or its design.

7.1.2. Inference Phase (INF)

Following the determination of the PLS model for the given dataset in CONF, INF carries out an operation of Markov chain Monte Carlo (MCMC) probabilistic inference to obtain a pairwise similarity matrix. This output matrix holds the information that will be used in determining the clusters of entities. As input, INF takes the dataset and the PLS model as determined by CONF. As output, it produces the matrix of posterior pairwise probabilities. To generate this matrix, INF runs an MCMC sampling operation using four parameters: the number of chains, the number of iterations in each chain, the number of iterations to be skipped, and the initial values for hyper-parameters. Following the MCMC run, the pairwise similarity matrix is computed by taking averages over all non-skipped iterations over all chains.

7.1.3. Clustering Phase (CLUS)

After obtaining a pairwise similarity matrix by probabilistic inference, we still have to determine the exact clusters of entities and apply hypothesis-testing to detect the significant GO terms associated with those clusters, if applicable. CLUS is the phase that takes this matrix and applies a two-stage clustering operation to obtain clusters (subsets) of genes. This operation has two parameters as input: merge threshold and extension threshold, which are used in its two stages. Two-stage clustering may

result in different numbers of overlapping or non-overlapping clusters depending on the given thresholds and the similarity matrix. The threshold parameters determine the stringency of the operation; larger thresholds will result in a larger number of clusters with fewer members, representing finer similarity relations, whereas smaller thresholds will result in a smaller number of clusters that represent coarser similarity relations. The resulting clusters are then received by the evaluation phase for hypothesis testing.

7.1.4. Evaluation Phase (EVAL)

Multiple hypothesis testing is applied on the clusters in EVAL. This operation requires GO term assignments for all genes and an alpha parameter (a significance threshold) to use in hypothesis testing. For any given cluster, all GO terms that are directly or indirectly annotated with its member genes are considered as possible hypotheses. Each of these GO terms belongs to one of the three categories: cellular component, molecular function, or biological process. EVAL applies multiple hypothesis testing with Bonferroni or Benjamini-Hochberg correction for multiple testing to the whole set of hypotheses comprised of GO terms from all three categories, and the resulting significant GO terms associated with each cluster are reported in the final output.

7.2. Implementation

7.2.1. CONF: Temporal segmentation (TS)

TS is a simple operation where segments of time points that display a correlated behaviour are discerned by applying hierarchical agglomerative clustering to the vectors of values over all entities at each time point in the dataset. The resulting dendrogram is divided by a selected segmentation threshold, and the resulting subtrees are marked as the time segments of the piecewise linear sequence model that will be used in the next phase. The PLS segmentation can also be set manually by the user (Figure 1).

7.2.2. INF: MCMC for IMPLS

CLUSTERnGO (CnG) models time-course profiles using an infinite mixture of piecewise linear sequences (IMPLS). To compute the posterior of IMPLS, it uses an MCMC procedure.

7.2.2.1. The IMPLS model. Suppose that we have N entities indexed by $i \in \{1, ..., N\}$ and their profiles x_i, vectors of size M, which are to be modelled as distributed around an unknown number of piecewise linear sequences. Cluster Mixture component assignments zi of these entities are assumed to come from a two-parameter CRP, an iterative construction for a PYP:

$$z_{1:N} \mid \alpha, d \ \sim \ CRP(\alpha, d) \tag{7.1}$$

A PLS model is defined by L parameters in the following order: initial value, slope of the first segment, jump to the second segment, slope of the second segment, jump to the third segment, and so on. The prior variances of these three types of parameters are given by $V_{\text{init}}, V_{\text{jump}}, V_{\text{slope}}$. These variances form the diagonal of the matrix Σ_μ. For every cluster mixture component $k \in \{1, ..., K\}$ there is an L-vector μ_k that defines a PLS with a Gaussian prior:

$$\mu_k \mid V_{\text{init}}, V_{\text{jump}}, V_{\text{slope}} \ \sim \ \mathcal{N}(\mu_k \mid 0, \Sigma_\mu) \tag{7.2}$$

Each cluster mixture component also has a precision (inverse variance) parameter λ_k with a Gamma prior:

$$\lambda_k \mid a, b \ \sim \ \mathcal{G}(\lambda_k \mid a, b) \tag{7.3}$$

Finally, we have the likelihood, which determines that each time-series is distributed according to a Gaussian with mean $C\ \mu_k$ and variance $1/\lambda_k$, where k is the cluster mixture component that this sample belongs to. C is a constant matrix that is either

manually specified or determined semi-automatically by the CONF procedure. This matrix transforms PLS parameters μ_k into the mixture component mean:

$$x_i \mid \mu, \lambda, z_i \sim \prod \mathcal{N}(x_i \mid C\ \mu_k, \lambda_k^{-1} I) \tag{7.4}$$

C is a matrix of basis vectors and each time-series (here, simply a finite dimensional vector) is modelled by $x = C\ \mu + \epsilon$. Mean μ is zero, $< \mu >= 0$. The covariance of x is thereby $< xx' >= C < \mu\mu' > C' + R = CC' + R$. The matrix C is constructed such that typical x are Piecewise Linear Sequences, such sequences will have the conditional covariance $CC' + R$.

7.2.2.2. MCMC inference. A special Markov Chain Monte Carlo (MCMC) procedure was adopted in the analysis of this model due to the presence of matrix C, which transforms the parameter vector. We run Metropolis-Hastings (MH) steps to sample the cluster mixture component precisions λ_k and use these values to run collapsed Gibbs sampling steps to sample zi by integrating out the cluster mixture component centres μ_k. Our MCMC algorithm consists of three steps repeatedly applied to converge to the target distribution $p(x, z, \lambda, \alpha, d, a, b)$.

(i) For each $k = 1 \ldots K$, apply MH steps to re-sample λ_k by $p(\lambda_k | x_{1:N}, z_{1:N})$.

(ii) For each $i = 1 \ldots N$, apply collapsed Gibbs sampling for z_i by $p(z_i | x_{1:N}, z_{-i}, \lambda_{1:K})$ using auxiliary variable method for sampling new λ_k.

(iii) Apply MH steps to sample the hyper-parameters; α, d, a, b by their, respective, non-informative priors $1/\alpha$, $1/d$, 1 and b.

The PLS prior parameters $V_{\text{init}}, V_{\text{jump}}, V_{\text{slope}}$ are each fixed at a sufficiently large number to assign equal probabilities for different PLS parameter values. The user is allowed to interact with the MCMC on the initial values for the IMPLS hyper-parameters, the number of iterations to be carried out, the number of chains, or the skip value. The default values for the number of iterations to be carried out, the number of chains, and the skip value for the burn-in period were set as 10000, 20

and 2500, respectively. The number of iterations and the number of chains are kept at high values to help the MCMC inference to more closely approach its stationary distribution. In practice, this enables the INF phase to yield very similar results in sucessive runs, even though it is based on a probabilistic algorithm. The default initial settings for the hyper-parameters are as follows; a = 2.1, b = 0.24, d = 0.001 and alpha = 100 although these parameters are readjusted during the iterations.

7.2.3. CLUS: Two-stage clustering (TSC)

CLUS is a deterministic phase where decisions are based on simple numerical comparisons on pairwise posterior probabilities. Clusters cannot be determined in the INF phase, because data is finite and there is uncertainty in the infinite mixture posterior. The CLUS phase operates on this posterior to decide on the final clusters. The inference results contained in the pairwise similarity matrix are translated into a set of clusters that indicate groups of related entities through the application of a two-stage operation in the clustering phase. The degree of similarity in clustering is determined by two parameters: the merge threshold and the extension threshold.

Let M be the pairwise similarity matrix where Mij denotes the similarity between entity i and entity j, namely, the posterior pairwise probabilities between these entities as obtained from MCMC. Given this matrix M and the two threshold parameters, two-stage clustering runs as follows:

(i) Prepare an initial set Π of 1-element clusters.
(ii) Choose the cluster pair (S_a, S_b) where the minimum similarity value between any $i \in S_a$ and $j \in S_b$ is the maximum among cluster pairs.
(iii) Remove S_a and S_b, and insert their union $S_a \cup S_b = S_c$ into the set of clusters; Π.
(iv) Continue from step 2 until the obtained similarity between $i \in S_a$ and $j \in S_b$ is smaller than the merge threshold.
(v) Choose the cluster-entity pair (S_a, j) where the minimum similarity value between any $i \in S_a$ and j is the maximum among all pairs.
(vi) Remove S_a and insert its increment $S = S_a \cup \{j\}$ into the set of clusters; Π.

(vii) Continue from step 5 until the obtained similarity between $i \in S_a$ and j is smaller than the extension threshold.

Among these steps, 2, 3, and 4 designate the first stage where small clusters are merged into larger clusters, and 5, 6, and 7 designate the second stage where clusters are further extended by inserting elements. Intuitively, the merge threshold determines the size of cluster cores, whereas the extension threshold determines the extent of overlap among cluster peripheries. Lowering the extension threshold in stage 2 can result in wide cluster peripheries that overlap for many genes. Lowering the merge threshold in stage 1 will yield few large cluster cores, thereby effectively constraining the possibilities of overlaps in stage 2. Using this methodology, there is no need for any a priori knowledge or assumption concerning the number of clusters that will be identified at the end of the process. The default settings for the merge and the extension threshold parameters were both 0.5, although they can be individually set by the user to any value between 0 and 1.

7.2.4. EVAL: Multiple hypothesis testing

The identified clusters of genes are significantly associated with a biological ontology through the application of multiple hypothesis testing in EVAL. Gene Ontology (GO), where each gene is annotated by a list of terms from three domains: cellular component, molecular function, and biological process was adopted as the biological ontology in this analysis (Ashburner et al., 2000). To determine if a given cluster is annotated by a given GO term at a frequency greater than by chance, the p-value is computed using the hypergeometric distribution:

$$P = 1 - \sum_{i=0}^{k-1} \frac{\binom{M}{i}\binom{N-M}{n-i}}{\binom{N}{n}} \tag{7.5}$$

Here, N is the total number of unique genes, M is the number of genes annotated by the term, n is size of the cluster, and k is the number of annotated genes in the cluster. Bonferroni correction was used as a conservative action to control the family-wise error

rate. Although the Bonferroni correction is set as default, the Benjamini-Hochberg procedure is also provided as a more relaxed option to control the false discovery rate (FDR) at level alpha. The assigned GO term is identified as significant if the p-value is less than the significance threshold, whose default was set as $\alpha = 0.01$.

8. CONCLUSION

Let us summarize the contributions of this thesis and point out some possible future directions. As listed at the end of the Introduction, these contributions are:

1) Infinite Multiway Mixture with Factorized Latent Parameters (IMM)
2) Infinite Mixture of Piecewise Linear Sequences (IMPLS)
3) Cumulative Statistics and Entropy Agglomeration (EA)
4) Clustering Words by Entropy Agglomeration
5) CLUSTERnGO (CnG): Cluster analysis of gene expression profiles

Let us now briefly describe the listed contributions in the context of their relationship. First of all, the last one in the list, CnG, serves as the backbone of the ideas developed in the thesis. It is a gene expression analysis application that conducts its operations with a structured pipeline. The design of CnG presents a basic outline that can serve as an example for future applications. As we state in the Abstract and in the beginning of the Introduction, the four phases of CnG (*Configuration, Inference, Clustering, Evaluation*) represent a sufficiently general conceptual relation, so that the specific implementation of each phase can be replaced by an alternative approach without requiring modifications on this general structure. In its present implementation, the first two phases of CnG implement an MCMC inference method that deploys an IMPLS model. The third phase implements a heuristic clustering algorithm (TSC) and the fourth phase applies multiple hypothesis testing on the resulting clusters with respect to Gene Ontology terms.

The first two of the remaining four contributions, 1 and 2, develop two Bayesian nonparametric models, IMM and IMPLS, respectively. IMM is a simple but powerful formulation of a generative model that combines infinite mixture modeling with tensor factorization. The index notation borrowed from tensor factorization is used to formulate a multiway mixture model over several dimensions. IMM provides an elegant model that explores certain theoretical ideas from related domains. IMPLS, on the

other hand, is a model specifically designed according to the needs that originated from our task to analyze gene expression profiles. It formulates an infinite mixture model that presupposes a structure called a Piecewise Linear Sequence (PLS) for each of its mixture components. An MCMC inference method for IMPLS was developed and was implemented as part of CnG.

The remaining two contributions, 3 and 4, represent an independent study that originated from a research question that emerged in the development of CnG. It is a question that concerns the third phase of CnG, where the application needs to take the combinatorial sample set obtained from MCMC inference, and analyze it to decide on a final set of clusters that will then be sent to Evaluation, the fourth and final phase. It is a very fundamental question: How can we express and summarize partitionings and feature allocations? The heuristic solution implemented in CnG, TSC, despite being very useful as a practical solution, does not provide an adequate answer in face of the theoretical question.

So in these two contributions, 3 and 4, we formulate a methodology that we call *cumulative statistics*, develop a clustering algorithm that we call Entropy Agglomeration (EA) and apply EA on a literary text, namely Ulysses (1922) by James Joyce. We believe that the formulation of *cumulative statistics* is self-explanatory and that the evidence presented by the experiment is sufficient to say that EA is a useful algorithm.

Having described the contributions in their relationship, let us now point out some possible directions for future research that can proceed from the ideas presented in the thesis. CnG (Standard version: CnG v0.30) is a very structured application, and all of its phases are open to advancements specific to their own functionality. In the first two phases, IMPLS can be replaced by any other Bayesian generative model and MCMC can be replaced by any other inference methodology for that model. We have already developed additional functionality to enable phenotype analyses in the fourth phase, and included this functionality in the new version, CnG v0.31. Another work to do is to replace the heuristic TSC algorithm in the Clustering phase of CnG with the Entropy Agglomeration algorithm that we have developed.

In concluding the thesis, let us ask the basic question concerning our work: Do the contributions we present in the thesis measure up to our initial objectives? In other words, were we able to present sufficient tools and techniques in this thesis for researchers in bioinformatics to be able to extract structural information effectively and to detect gene interactions reliably from gene expression datasets? The short and simple answer is: Yes, CnG was actually used to produce novel biological insights from experiments with yeast and mouse cells, and the evidence of these accomplishment were documented in our articles in *Bioinformatics* and *Stem Cells* [52, 53].

But according to the long answer, for a variety of reasons, these contributions cannot be considered to constitute an ultimate solution that is fully satisfactory with respect to our initial objective to extract reliable structural information from noisy datasets. First of all, what we develop here are mere models inherently abstracted from application concerns, that is, even if we take particular application goals into account in the design process, the formulations as they are finally presented in the thesis are indifferent to external evaluation by their nature and by definition. Secondly, what was involved in this work was fundamentally an endeavour to construct tools: an effort to devise coherent formulations and build useful frameworks. As a result, whether these tools will be able to meet the application needs is highly dependent on their actual and potential users. Thirdly, we acknowledge the fact that, since we had to deal with difficult theoretical problems in this work, the mathematical connections of the presented models have remained somehow fuzzy. Nevertheless, fourthly and finally, the experiments we conducted with our tools left us convinced that these tools are and will be useful in the present and future approaches to the clustering problem. So even though we cannot claim a decisive triumph with respect to our initial objective, we think that the work we present here offers a safe ground with a sound direction.

To sum up, we think that CnG, both as an actual application and as a conceptual apparatus, can serve as a starting point for several directions of research. We also speculate that the statistical methodology that we introduce as *cumulative statistics* opens up a useful new avenue for statistical formulations of combinatorial objects.

REFERENCES

1. Ferguson, T. S., "A Bayesian analysis of some nonparametric problems", *The Annals of Statistics*, Vol. 1, No. 2, p. 209–230, 1973.

2. Teh, Y. W., "Dirichlet Processes", C. Sammut and G. I. Webb (Editors), *Encyclopedia of Machine Learning*, Springer-Verlag, Berlin, 2010.

3. Ewens, W. J., "The sampling theory of selectively neutral alleles", *Theoretical Population Biology*, Vol. 3, No. 1, pp. 87–112, 1972.

4. Watterson, G. A., "The sampling theory of selectively neutral alleles", *Advances in Applied Probability*, Vol. 6, No. 3, pp. 463–488, 1974.

5. Kingman, J. F. C., "Random Partitions in Population Genetics", *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, Vol. 361, No. 1704, pp. 1–20, 1978.

6. Andrews, G. E., *The theory of partitions*, Addison-Wesley Publications, Reading, Mass., 1976.

7. Pitman, J., "Combinatorial Stochastic Processes", *Lecture Notes in Mathematics, Vol. 1875*, Springer-Verlag, Berlin, 2006.

8. Bishop, C. M., *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer, 1st ed. 2006. corr. 2nd printing edn., October 2007.

9. Neal, R. M., "Bayesian mixture modeling", C. R. Smith, G. J. Erickson and N. P. O. (Editors), *Maximum Entropy and Bayesian Methods: Seattle, 1991*, Vol. 11, pp. 197–211, Kluwer Academic Publishers, The Netherlands, 1992.

10. Blackwell, D. and J. B. MacQueen, "Ferguson Distributions Via Polya Urn

Schemes", *The Annals of Statistics*, Vol. 1, No. 2, pp. 353–355, 1973.

11. Neal, R. M., "Markov Chain Sampling Methods for Dirichlet Process Mixture Models", *Journal of Computational and Graphical Statistics*, Vol. 9, No. 2, pp. 249–265, 2000.

12. Teh, Y. W., "Dirichlet Processes", *Encyclopedia of Machine Learning*, Springer, 2010.

13. Shashua, A., R. Zass and T. Hazan, "Multi-way clustering using super-symmetric non-negative tensor factorization", *In Proc. of the European Conference on Computer Vision (ECCV)*, 2006.

14. Zhou, D., J. Huang and B. Schölkopf, "Learning with Hypergraphs: Clustering, and Classification, Embedding", J. P. B. Schölkopf and T. Hoffman (Editors), *Advances in Neural Information Processing Systems 19*, pp. 1601–1608, MIT Press, 2007.

15. Banerjee, A., S. Basu and S. Merugu, "Multi-Way Clustering on Relation Graphs", *In Proc. SIAM Conf. Data Mining*.

16. Yilmaz, K. and A. T. Cemgil, "Probabilistic Latent Tensor Factorisation", *In Proc. of International Conference on Latent Variable analysis and Signal Separation*, pp. 346–353, 2010.

17. Meeds, E. and S. Roweis, *Nonparametric Bayesian Biclustering*, Tech. Rep. UTML TR 2007–001, Department of Computer Science, University of Toronto, Toronto, Canada, June 2007.

18. Neal, R. M., "Markov chain sampling methods for Dirichlet process mixture models", *Journal of Computational and Graphical Statistics*, Vol. 9, p. 249–265, 2000.

19. Yeung, K. Y., C. Fraley, A. Murua, A. E. Raftery and W. L. Ruzzo, "Model-based

clustering and data transformations for gene expression data", *Bioinformatics*, Vol. 17, pp. 977–987, 2001.

20. Eisen, M. B., P. T. Spellman, P. O. Brown and D. Botstein, "Cluster analysis and display of genome-wide expression patterns", C. R. Smith, G. J. Erickson and P. O. Neudorfer (Editors), *Proceedings of the National Academy of Sciences*, Vol. 95, pp. 14863–14868, 1998.

21. Tamayo, P., D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander and T. R. Golub, "Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation", *Proc. Natl Acad. Sci. USA*, Vol. 96, pp. 2907—-2912, 1999.

22. Tavazoie, S., J. D. Hughes, M. J. Campbell, R. J. Cho and G. M. Church, "Systematic determination of genetic network architecture", *Nat. Genet.*, Vol. 22, pp. 281—285, 1999.

23. Medvedovic, M. and S. Sivaganesan, "Bayesian infinite mixture model based clustering of gene expression profiles", *Bioinformatics*, Vol. 18, p. 1194–1206, 2002.

24. Qin, Z. S., "Clustering microarray gene expression data using weighted Chinese restaurant process", *Bioinformatics*, Vol. 22, pp. 1988—-1997, 2006.

25. Joshi, A., Y. V. Peer and T. Michoel, "Analysis of a Gibbs sampler for model based clustering of gene expression data", *Bioinformatics*, Vol. 24, pp. 176–183, 2008.

26. Dikicioglu, D., E. Karabekmez, B. Rash, P. Pir, B. Kirdar and S. G. Oliver, "How yeast re-programmes its transcriptional profile in response to different nutrient impulses", *BMC Systems Biology*, Vol. 148, No. 5, 2011.

27. Liu, J. S., *Monte Carlo Strategies in Scientific Computing*, Springer, New York, 2002.

28. Fidaner, I. B. and A. T. Cemgil, "Summary Statistics for Partitionings and Feature Allocations", *Advances in Neural Information Processing Systems*, Vol. 26, 2013.

29. Kingman, J. F. C., *Poisson processes*, Oxford University Press, Oxford, 1992.

30. Pitman, J. and M. Yor, "The two-parameter Poisson–Dirichlet distribution derived from a stable subordinator", *Annals of Probability*, Vol. 25, No. 2, pp. 855–900, 1997.

31. Sethuraman, J., "A constructive definition of Dirichlet priors", *Statistica Sinica*, Vol. 4, pp. 639–650, 1994.

32. Meeds, E., Z. Ghahramani, R. M. Neal and S. T. Roweis, "Modeling Dyadic Data with Binary Latent Factors", *Advances in Neural Information Processing Systems 19*, 2007.

33. Teh, Y. W., M. I. Jordan, M. J. Beal and D. M. Blei, "Hierarchical Dirichlet processes", *Journal of the American Statistical Association*, Vol. 101, No. 476, p. 1566–1581, 2006.

34. Griffiths, T. L. and Z. Ghahramani, "The Indian buffet process: An introduction and review", *Journal of Machine Learning Research*, Vol. 12, p. 1185–1224, 2011.

35. Broderick, T., J. Pitman and M. I. Jordan, "Feature Allocations, Probability Functions, and Paintboxes", *Bayesian Analysis*, Vol. 8, No. 4, pp. 801–836, 2013.

36. Teh, Y. W., C. Blundell and L. T. Elliott, "Modelling genetic variations with fragmentation-coagulation processes", *Advances in Neural Information Processing Systems 23*, 2011.

37. Orbanz, P. and Y. W. Teh, "Bayesian Nonparametric Models", C. Sammut and G. I. Webb (Editors), *Encyclopedia of Machine Learning*, Springer-Verlag, Berlin, 2010.

38. Medvedovic, M., K. Yeung and R. Bumgarner, "Bayesian mixture model based clustering of replicated microarray data", *Bioinformatics*, Vol. 20, p. 1222–1232, 2004.

39. Liu, X., S. Sivanagesan, K. Yeung, J. Guo, R. E. Bumgarner and M. Medvedovic, "Context-specific infinite mixtures for clustering gene expression profiles across diverse microarray dataset", *Bioinformatics*, Vol. 22, pp. 1737–1744, 2006.

40. Nemenman, I., F. Shafee and W. Bialek, "Entropy and inference, revisited", *Advances in Neural Information Processing Systems 14*, 2002.

41. Shannon, C. E., "A Mathematical Theory of Communication", *Bell System Technical Journal*, Vol. 27, No. 3, p. 379–423, 1948.

42. Archer, E., I. M. Park and J. W. Pillow, "Bayesian Entropy Estimation for Countable Discrete Distributions", *Journal of Machine Learning Research*, Vol. 15, pp. 2833–2868, 2014.

43. Simovici, D., "On Generalized Entropy and Entropic Metrics", *Journal of Multiple Valued Logic and Soft Computing*, Vol. 13, No. 4-6, pp. 295–320, 2007.

44. Ellerman, D., "Counting distinctions: on the conceptual foundations of Shannon's information theory", *Synthese*, Vol. 168, No. 1, pp. 119–149, 2009.

45. Fisher, R. A., "The use of multiple measurements in taxonomic problems", *Annals of Eugenics*, Vol. 7, No. 2, pp. 179–188, 1936.

46. Ideker, T., V. Thorsson, J. A. Ranish, R. Christmas, J. Buhler, J. K. Eng, R. Bumgarner, D. R. Goodlett, R. Aebersold and L. Hood, "Integrated genomic and proteomic analyses of a systematically perturbed metabolic network", *Science*, Vol. 292, No. 5518, pp. 929–934, 2001.

47. Pevehouse, J. C., T. Nordstrom and K. Warnke, "The COW-

2 International Organizations Dataset Version 2.1", *Conflict Management and Peace Science*, Vol. 21, No. 2, pp. 101–119, 2004, http://www.correlatesofwar.org/COW2%20Data/IGOs/IGOv2-1.htm, accessed at March 2016.

48. Wood, F., J. Gasthaus, C. Archambeau, L. James and Y. W. Teh, "The Sequence Memoizer", *Communications of the ACM*, Vol. 54, pp. 91–98, 2011.

49. Teh, Y. W., "A hierarchical Bayesian language model based on Pitman-Yor processes", *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (ACL-44)*, pp. 985–992, 2006.

50. Fidaner, I. B. and A. T. Cemgil, "REBUS: entropy agglomeration of text", http://cmpe.boun.edu.tr/content/REBUS, 2014, accessed at March 2016.

51. Fidaner, I. B., A. Cankorur-Cetinkaya, D. Dikicioglu, B. Kirdar, A. T. Cemgil and S. G. Oliver, "CLUSTERnGO (CnG) Software", http://cmpe.boun.edu.tr/content/CnG, 2015, accessed at March 2016.

52. Fidaner, I. B., A. Cankorur-Cetinkaya, D. Dikicioglu, B. Kirdar, A. T. Cemgil and S. G. Oliver, "CLUSTERnGO: A user-defined modelling platform for two-stage clustering of time-series data", *Bioinformatics*, 2016.

53. Mulvey, C. M., C. Schröter, L. Gatto, D. Dikicioglu, I. B. Fidaner, A. Christoforou, M. J. Deery, L. T. Cho, K. K. Niakan, A. Martinez-Arias and K. S. Lilley, "Dynamic Proteomic Profiling of Extra-Embryonic Endoderm Differentiation in Mouse Embryonic Stem Cells", *Stem Cells*, 2015.