Published in Image Processing On Line on YYYY–MM–DD.
Submitted on YYYY–MM–DD, accepted on YYYY–MM–DD.
ISSN 2105–1232 © YYYY IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol

Clustering Multivariate Ordinal Data

Thomas Michel[1], Ali Ramlaoui[2], Théo Rudkiewicz[3]

[1 2 3] ENS Paris-Saclay, France
[1] (thomas.michel1@ens-paris-saclay.fr)
[23] (firstname.lastname@ens-paris-saclay.fr)

PREPRINT July 21, 2024

Contents

1 Introduction **2**

2 Method **4**
2.1 AECM algorithm. 4
2.2 Univariate model 5
2.2.1 Notations and goal 5
2.2.2 Optimization 6
2.3 Stochastic Binary Ordinal Search 7
2.3.1 Probabilistic model 7
2.3.2 Parameter estimation 8
2.4 Globally Ordered Data model 9
2.4.1 Probabilistic model 9
2.4.2 Parameter estimation 10

3 Experiments **11**
3.1 Synthetic data 11
3.2 Real-life datasets 13
3.2.1 Datasets 13
3.2.2 Evaluation method. 14
3.2.3 Experiments with real-life datasets 15

4 Conclusion **15**

A Synthetic data **18**

B Real-life datasets **20**

C Generic proofs **24**
C.1 Ternary search algorithm . . . 24
C.2 Concavity . . . 25

D BOS Model proofs **25**
D.1 Notations . . . 25
D.2 Polynomiality . . . 25
D.3 Concavity . . . 27
D.4 Efficient computation of the likelihood . . . 28
D.4.1 Mathematical proofs . . . 28
D.4.2 Algorithm . . . 31

E GOD Model proofs **32**
E.1 Probabilistic model . . . 32
E.2 Likelihood expression . . . 33
E.3 Concavity . . . 34
E.4 Algorithm . . . 35

Abstract

We introduce a novel algorithm for clustering multivariate ordinal data, building upon the work of Biernacki and Jacques [2016]. We examine two processes for generating univariate ordinal data and present an algorithm for efficiently estimating the parameters of these processes. For the first and pre-existing process known as Binary Ordinal Search, we propose a polynomial algorithm for parameter estimation which improve the previously proposed exponential algorithm. For the second process, we propose an alternative model with similar properties, called Globally Ordered Data. Finally, we utilize the AECM algorithm to cluster multivariate ordinal data based on these models and showcase the efficiency of our algorithm on synthetic and real data.

Source Code

We provide a Python implementation of all the algorithms presented in this article. The source code is available from the github repository of this article[1]. This includes the estimation algorithm for the BOS and GOD models and the AECM algorithm for clustering multivariate ordinal data. We also provide a Jupyter notebook to reproduce the experiments presented in this article.

Keywords: Ordinal data, Clustering, AECM algorithm, Parameter estimation

1 Introduction

The exploration of hidden structures within datasets is a crucial task for data scientists, and clustering serves as a valuable tool for this. Mixture models have emerged as a standard approach for clustering due to their capacity to provide a well-defined mathematical framework for parameter estimation and model selection. These models, instrumental in determining the number of clusters, not only encapsulate classical geometric methods but also find successful application in diverse practical scenarios.

In the realm of model-based clustering, the classification of data hinges on the availability of a suitable probability distribution tailored to the nature of the data at hand—be it numerical, rankings, functional, or categorical. Notably, ordinal data, where categories possess a specific order, represent a common occurrence, especially in fields like marketing where product evaluations are solicited through

[1] https://github.com/Thomick/Ordinal-data-clustering

ordinal scales. Despite their prevalence, ordinal data have received comparatively less attention in the context of model-based clustering. Often, practitioners resort to transforming ordinal data into quantitative or nominal formats to align with readily applicable distributions, neglecting valuable order information.

This paper delves into the less-explored domain of model-based clustering for ordinal data, specifically focusing on ordinal data derived from sample surveys. Ordinal data find widespread application in fields such as social sciences, psychology, marketing, healthcare, and more. They enable researchers to capture nuanced information, such as preferences, attitudes, or severity levels, in cases where continuous measures are neither significant nor possible. For example, when assessing tumor severity, the precise size may not be as crucial as the current state of development of the disease as assessed by specialists. The use of ordinal data enriches the comprehension of subjective opinions, behaviors, and hierarchical relationships across diverse research contexts.

Multiple approaches have been proposed to model ordinal data. The first primary approach involves modeling a function of the cumulative probabilities of the ordinal categories as a linear function of some covariates. A comprehensive review of these models can be found in Agresti [2010]. The second main approach involves modeling the ordinal categories as a function of a latent continuous variable, which is then linked to the ordinal categories through a link function. We assume that the ordinal observations are generated by a form of discretization of the latent continuous variable. An illustrative example of this approach is the ordered probit model, a generalization of the probit model for ordinal data.

A final approach that will particularly interest us in this paper is the use of a probabilistic model that directly captures the data-generating process for ordinal data, ensuring it exhibits desired properties. Two main models have been proposed in this category. The first and most studied one is the CUB model and its extensions [D'Elia and Piccolo, 2005], which suggests modeling the data-generating process as a mixture of uniform and binomial distributions. Later, an additional Dirac distribution was introduced in the mixture, removing the dependence on the distance between categories imposed by the binomial distribution with the nonlinear CUB model [Manisera and Zuccolotto, 2014]. Another notable model in this category is the Binary Ordinal Search model, proposed by Biernacki and Jacques [2016]. This model posits that the observed data is the outcome of a stochastic process involving a binary search algorithm with corrupted comparisons. Parameterized with a position parameter (modal category) and a precision parameter, this model exhibits desirable properties similar to those of the CUB model. These properties include a unique mode, a probability distribution decrease on either side of the mode, and the flexibility to accommodate uniform or Dirac distributions.

In the context of clustering, a common approach is to use a mixture model for clustering the data. The mixture model is a probabilistic model that assumes the data is generated by a convex combination of several probability distributions. The clustering process then assigns each data point to the most likely distribution component given the data. In the case of the BOS model, maximum likelihood estimation using an EM algorithm can be employed, leveraging the latent variable interpretation of the binary search algorithm. While combinatorial complexity limits straightforward estimation for models based on latent Gaussian variables, the proposed approach remains tractable for ordinal data with up to eight categories—a common scenario for most ordinal variables. By contrast, the CUB model cannot be used for clustering in this way, as two CUB distributions cannot be distinguished from one another due to the uniform distribution component.

In this paper, we replicate and extend the findings of Biernacki and Jacques [2016]. We re-implemented their suggested probabilistic model, parameter estimation method, and model-based clustering algorithm in Python. Taking inspiration from their approach, we propose an alternative probabilistic model with similar properties. The aim is to address computational limitations, enabling the clustering of more extensive datasets with more categories than what the previous method allows. We also present an analysis of this new method to justify the decreased computational cost of

estimating parameters for this model. Additionally, we test Biernacki and Jacques [2016]'s approach on real-world datasets and compare it to the proposed approach, along with baseline models. This is conducted on different datasets of multiple natures to check whether the proposed methods are successful in these practical settings and to identify their advantages. The ultimate goal is to assess whether the gains are significant against methods not adapted for ordinal datasets, deciding whether these approaches are interesting to use as a default method of choice for this type of variable.

2 Method

Our goal is to cluster multivariate data. Each dimension contains data generated by a random process characterized by parameters dependent on the cluster. To estimate the cluster, one can utilize the AECM algorithm, as introduced in [Meng and Van Dyk, 1997]. This algorithm is quite versatile, requiring only the ability to estimate the univariate parameters of a weighted set of data points. We present this algorithm in section 2.1.

Similar to the approach in Biernacki and Jacques [2016], we concentrate on ordinal data. Therefore, we assume that each dimension follows a process that generates ordinal categories. In section 2.2, we introduce two random processes for modeling ordinal data: the binary ordinal search (BOS) model from Biernacki and Jacques [2016] and the globally ordered data (GOD) model.

To apply the proposed methods, each dimension of the data should represent an ordinal category. These categories must adhere to the following properties:

- The categories must be well-ordered (ordinal data): They are linearly ordered, and each non-empty subset contains a least element. This implies that any element can be compared to any other, and we can enumerate all the categories in increasing order.
- The set of categories is finite. This simplifies the previous assumption to the existence of a linear ordering. This assumption is necessary to ensure that the stochastic search terminates after a fixed number of steps, implying a finite number of possible runs of the search.

2.1 AECM algorithm.

Similarly to the EM algorithm, Alternating Expectation-Conditional Maximization (AECM) [Meng and Van Dyk, 1997] is separated in two steps. However, in this case, we consider multivariate ordinal data with different possible distributions (clusters) priors for the data. Similar to the Gaussian Mixture Model case, the algorithm use latent variables w_{ik} which describe whether the data x_i belongs to the cluster k or not, and parameters $(\alpha_k)_{k\in\{1,\dots,p\}}$ which describe the probability of belonging to each of the p cluster. We also note $\theta_k^{(t)}$ the parameters of the k-th cluster at the t-th iteration.

1. Expectation step: The expectation step consists in computing the probability for every data point to belong to each cluster:

$$\mathbb{P}(w_{ik} = 1 | x_i, \alpha^{(t)}, \theta^{(t)}) = \frac{\alpha_k^{(t)} \mathbb{P}(x_i | w_{ik} = 1, \theta_k^{(t)})}{\sum_{l=1}^{p} \alpha_l^{(t)} \mathbb{P}(x_i | w_{il} = 1, \theta_l^{(t)})}. \tag{1}$$

2. Maximization step: The parameters are updated using the new expected values for belonging to each cluster. Since there are two groups of latent parameters, the clusters weights $\alpha_k^{(t)}$ are updated first to maximize the log-likelihood:

$$\alpha_k^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{P}(w_{ik} = 1 | x_i, \alpha^{(t)}, \theta^{(t)}). \tag{2}$$

And then the parameters $(\theta_k^{(t+1)})$ are updated after using the internal parameters estimation for each cluster unsing the observations weighted by the probability of belonging to the cluster.

2.2 Univariate model

We now seek a random process to model univariate ordinal data among a finite number of categories. Assuming that we only concern ourselves with the order of the categories, we can, without loss of generality, consider our categories as $[\![1, m]\!]$ (when there are m categories), as long as the model does not utilize the distance between the numbers associated with two categories. Therefore, for the parameters θ of our model, the model gives $\forall i \in [\![1, m]\!], P(X = i|\theta)$ if X was generated from our random process.

As the goal is to represent data having a common distribution, we can suppose that there is an underlying true category $\mu \in [\![1, m]\!]$ and treat it as a parameter. Additionally, it is natural to include a precision parameter π, which correspond to how much noise is involved in the selection of a category. This holds true for both the BOS model and the GOD model.

In the following sections, we demonstrate how to estimate those parameters for a generic law. Later, we introduce the BOS model and explain how to apply this estimation technique. Likewise, we present how these techniques can be adapted in the context of the GOD model.

2.2.1 Notations and goal

Let's suppose we have a set of n independent observations $X = (x_i)_{i\in[n]}$, where $x_i \in [\![1, m]\!]$ follows a distribution P with parameters μ, π, where $\mu \in [\![1, m]\!]$ and $\pi \in [[a, b]]$. Our goal is to estimate μ and π by selecting the estimates that maximize the likelihood of the data.

$$(\mu, \pi) = \underset{(\mu,\pi)\in[\![1,m]\!]\times[a,b]}{\operatorname{argmax}} P(X|\mu, \pi)$$

As the data points are independent, we have:

$$P(X|\mu, \pi) = \prod_{i=1}^{n} P(x_i|\mu, \pi)$$

Since the number of possible values for x is finite, we can group the data by values and tally the occurrences of each value. Let n_i represent the number of occurrences of i in the data. We can write the likelihood as:

$$P(X|\mu, \pi) = \prod_{i=1}^{m} P(x_i|\mu, \pi)^{n_i}$$

In the AECM algorithm, each data point is assigned a weight w_i. As before, we can assume without loss of generality that we have only one observation of each value with a specific weight (we can always group the data by values and sum the weights of the observations). With the weights $W \in \mathbb{R}_+^n$, where w_i is the weight of the value i, we can express the weighted likelihood as:

$$P(W|\mu, \pi) = \prod_{i=1}^{m} P(i|\mu, \pi)^{w_i}$$

We can also write the weighted log-likelihood as:

$$L_W(\mu, \pi) := \log P(W|\mu, \pi) = \sum_{i=1}^{m} w_i \log P(i|\mu, \pi)$$

2.2.2 Optimization

To estimate μ and π, the approach suggested for the BOS model in Biernacki and Jacques [2016] is to employ the Expectation-Maximization algorithm. However, they highlight that it is simpler to initially estimate π for every possible value of μ and then estimate μ using the obtained π. The corresponding formulas are

$$\hat{\pi}_\mu = \underset{\pi \in [a,b]}{\operatorname{argmax}} P(W|\mu, \pi)$$

$$\hat{\mu} = \underset{\mu \in [\![1,m]\!]}{\operatorname{argmax}} \max_{\pi \in [a,b]} P(W|\mu, \pi) = \underset{\mu \in [\![1,m]\!]}{\operatorname{argmax}} P(W|\mu, \hat{\pi}_\mu).$$

Estimating π for a given μ is a one-dimensional optimization problem. We can utilize the EM algorithm for solving it, but it is also possible to employ a direct optimization algorithm, as we will demonstrate. For instance, if the function $\pi \mapsto L_W(\mu, \pi)$ is concave, we can use the ternary search algorithm to find the maximum.

Concavity of the weighted likelihood Suppose we have:

$$\forall x \in [\![1, m]\!], \pi \mapsto P(x|\mu, \pi) \text{ is log-concave}$$

Then, $\pi \mapsto L_W(\mu, \pi)$ is concave since a positive linear combination of concave functions is concave.

Ternary search algorithm The ternary search algorithm[2] can be employed to find the argmax of a unimodal or concave function within a given interval with a precision of ε in $\Theta(\log \frac{b-a}{\varepsilon})$ function evaluations. More precisely, in our case $(b - a \leqslant 1)$, it will require approximately 100 evaluations of the function for a precision of 10^{-10}. We will use this algorithm to estimate π for a given μ.

Evaluating the likelihood To execute the ternary search algorithm, we must efficiently evaluate the log-likelihood for a given π value. The log-likelihood is the sum of the logarithms of individual values' likelihoods, Therefore, the complexity of this task is $\Theta(mC_E(m))$ where $C_E(m)$ represents the complexity of calculating the likelihood for a single value of x.

For the BOS model, we observe that $\forall x \in [\![1, m]\!], \pi \mapsto P(x|\mu, \pi)$ is a polynomial of degree $\Theta(m)$ with coefficients depending on m, i, and μ. This observation is almost applicable to the GOD model as well. Since the function is polynomial, we can precompute these coefficients and then evaluate the likelihood of one value in $\Theta(m)$ operations. This results in a complexity of $\Theta(m^2)$ for evaluating the total likelihood of W for a given μ and π.

An essential consideration is that we've omitted the cost of precomputing the coefficients. While this cost is not necessarily negligible, unlike the likelihood evaluation, it occurs only once for a given m and can subsequently be applied across all iterations of the AECM algorithm and the ternary search algorithm.

Complexity Let n denote the number of observations, m represent the number of possible values for x, and $\varepsilon > 0$ indicate the precision of the estimate of π.

We can first group the data by values and sum the weights of the observations. This can be done in $\Theta(n)$ operations.

Next, to estimate π for a given μ, we can employ the ternary search algorithm. The number of iterations for the algorithm is $\Theta(\log \frac{b-a}{\varepsilon})$. In each iteration, computing the likelihood for two π

[2] https://en.wikipedia.org/wiki/Ternary_search

values incurs a complexity of $\Theta(\log \frac{b-a}{\varepsilon} C_E(m))$, where $C_E(m)$ denotes the complexity of computing the likelihood for a given π value.

Finally, to estimate μ, we must calculate the likelihood for every potential value of μ. This results in a total complexity of $\Theta(mC_E(m) \log \frac{b-a}{\varepsilon})$.

In our specific scenario, where $C_E(m) = \Theta(m^2)$ and $[a, b] \subset [0, 1]$, this yields a complexity of $\Theta(m^3 \log \frac{1}{\varepsilon})$ (excluding the precomputations of coefficients).

2.3 Stochastic Binary Ordinal Search

The BOS model is inspired by a standard binary search with added noise in the comparison. Consequently, the algorithm may at times misidentify the next subset for the search, ultimately causing it to overlook the sought-after value.

2.3.1 Probabilistic model

The stochastic binary ordinal search unfolds as follows: Let m be the number of categories. Then, for at most $m-1$ steps, we perform the following three operations. We start with the full set of categories, denoted as $e_1 = [\![1, m]\!]$. Then we perform the following steps:

At step j, we start with a subset of all the categories, denoted as $e_j = [\![l_j, u_j - 1]\!] \subseteq [\![1, m]\!]$.

1. Sample a breakpoint y_j uniformly in e_j ($y_j \sim \mathcal{U}(e_j)$).

2. Draw an accuracy indicator z_j from a Bernoulli distribution with parameter π ($z_j \sim \text{Bernoulli}(\pi)$). A value of $z_j = 1$ indicates that the comparison is perfect, and the next step will be computed optimally. A value of $z_j = 0$ implies a blind comparison at the next step.

3. Determine the new subset e_{j+1} for the next iteration. Firstly, split the subset into three intervals, namely $e_j^- = [\![l_j, y_j - 1]\!]$, $e_j^= = \{y_j\}$, and $e_j^+ = [\![y_j + 1, u_j - 1]\!]$. e_{j+1} will be chosen among these intervals. If the comparison is blind ($z_j = 0$), randomly select the interval with a probability proportional to its size. Alternatively, if $z_j = 1$ and the comparison is perfect, select the interval containing μ (or, by default, the one closest to it).

After $m-1$ steps, the resulting interval contains a single value, which is the observed result $e_m = \{x\}$ of the BOS model.

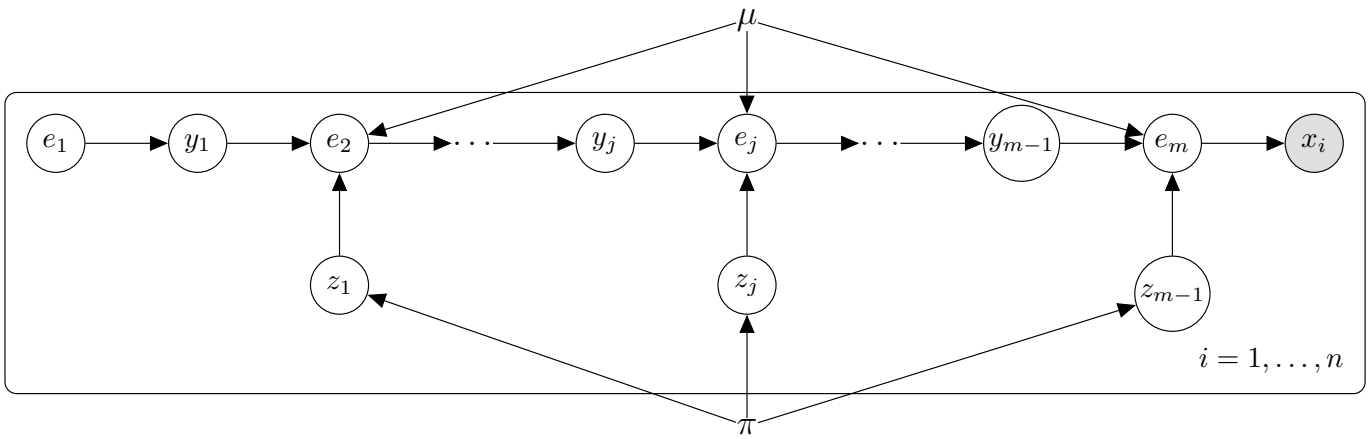


Figure 1: Graphical model of the stochastic Binary Ordinal Search.

2.3.2 Parameter estimation

We aim to estimate π and μ from a sample $(1, \dots, m)$ with weights $W \in \mathbb{R}_+^m$ generated by the GOD model. Our goal is to maximize the likelihood of the sample: $\Pr(W|\pi, \mu)$. We proceed as explained in section 2.2.2. All the proofs of this section are detailed in appendix D; here, we present only the main results.

Likelihood evaluation

Theorem 1 (Likelihood is polynomial)**.** $\forall m \in \mathbb{N}^*, \forall x \in [\![1, m]\!], \forall \mu \in [\![1, m]\!]$,*:*

$$\pi \mapsto \Pr(x|\mu, \pi)$$

is a polynomial function of degree at most $m - 1$.

Definition 1. *We denote by* $u[\mu, x, d]$ *the coefficient of the polynomial of degree* d *in the polynomial expansion of* $\pi \mapsto \Pr(x|\mu, \pi)$ *ie:*

$$\Pr(x|\mu, \pi) = \sum_{d=0}^{m-1} u[\mu, x, d]\pi^d$$

We demonstrate that the likelihood of a single observation is a polynomial function of degree at most $m-1$ in π. This result is significant because, once the coefficients are known, it implies that the likelihood of a single observation can be computed in $\Theta(m)$ operations, and the log-likelihood of a sample (which has at most m distinct values) in $\Theta(m^2)$ operations. For a fixed m, there are only m^2 polynomials to compute and store (one for each couple (x, μ)), leading to m^3 coefficients to store. We will show in the next paragraph 2.3.2 that these coefficients can be computed in $\mathcal{O}(m^5)$ operations.

Computing the coefficients

Definition 2. *To make it easier to understand the following results, we introduce a notation for the probability in the case of* h *categories* [3]*:*

$$l(x, \mu, h) := \Pr(x + 1|\mu + 1, \pi) \text{ with } h \text{ categories}$$

Theorem 2 (Computing the likelihood)**.** $\forall m \in \mathbb{N}^*, \forall x \in [\![1, m]\!], \forall \mu \in [\![1, m]\!], \forall \pi \in [0, 1]$*:*

$$\begin{aligned} l(x, \mu, h) = &\frac{1}{h} \sum_{y=x+1}^{h-1} l(x, \mu, y) \left[\left(\mathbb{1}\{\mu < y\} - \frac{y}{h} \right) \pi + \frac{y}{h} \right] \\ &+ \frac{1}{h} \left[(\mathbb{1}\{\mu = x \vee (x = 0 \wedge \mu \leqslant x) \vee (x = h - 1 \wedge \mu \geqslant x)\} - 1)\, \pi + \frac{1}{h} \right] \\ &+ \frac{1}{h} \sum_{y=0}^{x-1} l(x - y, \max(0, \mu - y), h - y) \left[\left(\mathbb{1}\{\mu > y\} - \frac{h - y - 1}{h} \right) \pi + \frac{h - y - 1}{h} \right] \end{aligned} \tag{3}$$

This theorem gives a recursive formula to compute the likelihood of a single observation. Using this formula, we can construct a dynamic programming algorithm to compute the coefficients of the polynomials. To do this, we proceed by increasing x, μ and h. We do directly the multiplication of the polynomials.

Each term of the sum requires the multiplication of a polynomial of degree at most $m - 1$ by a polynomial of degree 2 which gives a cost of $O(m)$. The sum has at most m terms, which gives a cost of $O(m^2)$. We need to apply this formula for each (x, μ, h) which gives a cost of $O(m^5)$.

[3] This notation may appear different from the one used in the appendix, but it is equivalent.

Log concavity

Conjecture 1 (Log concavity of the BOS model)**.** $\forall m \in \mathbb{N}^*, \forall x \in [\![1, m]\!], \forall \mu \in [\![1, m]\!]$*:*

$$\pi \mapsto \Pr(x|\mu, \pi)$$

is log-*concave on* $[0, 1]$.

This result is still a conjecture for $m > 94$ but has been verified for $m \leqslant 94$ almost formally. This is sufficient for almost all relevant data as it is quite rare to have many ordinal categories. We give more details on this verifications in appendix D. As explained in section 2.2.2, we directly observe that for all $\mu \in [\![1, m]\!]$, the function $\pi \mapsto L_W(\pi, \mu)$ is concave. Therefore, we can employ a ternary search algorithm to estimate π for a given μ.

Method efficiency As presented in section 2.2.2, we can estimate μ and π in $\mathcal{O}(m^3 \log \frac{1}{\varepsilon})$ operations once the coefficients u are computed. The coefficients u can be calculated in $\mathcal{O}(m^5)$ operations and stored in $\mathcal{O}(m^3)$ space. This represents a significant improvement compared to the EM algorithm proposed in Biernacki and Jacques [2016]. We provide a fully polynomial time algorithm with precision guarantees, while the EM algorithm offers no guarantees on precision, and the proposed algorithm becomes exponential in the number of categories (as demonstrated in section 3.1).

2.4 Globally Ordered Data model

The authors of Biernacki and Jacques [2016] motivated the use of binary search as such: "In order to minimize the number of potentially wrong comparisons, it is necessary to minimize the number of comparisons performed during the search process." However, we believe that minimizing the number of incorrect comparisons may not be an adequate intuition, and it is more crucial to minimize the probability of making a wrong guess. Motivated by this perspective, we have developed an alternative model where the data is compared with each category with some noise. We will refer to this model as the Globally Ordered Data (GOD) model.

We still consider a search for the parameter μ among the ordered categories. However, instead of conducting a binary search, we compare each category to the parameter μ and with probability π we get the correct answer. After making all these comparisons, we then select the category that corresponds to the minimum number of comparison error. This approach displays similar properties to the BOS model such as a unique mode, probability distribution decrease on either side of the mode, and the flexibility to accommodate uniform or Dirac distribution.

2.4.1 Probabilistic model

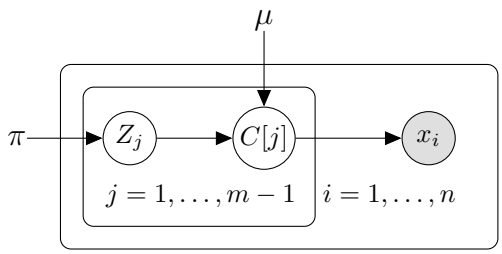


Figure 2: Graphical model of the GOD model.

The GOD model for m categories is characterized by two parameters $\mu \in [\![1, m]\!], \pi \in]\frac{1}{2}, 1]$. The observed data is only the selected category x. The latent variables are the vector $Z = (Z_1, ..., Z_{m-1}) \in$

$\{0,1\}^{m-1}$ and $C \in \{0,1\}^{m-1}$. $(Z_j)_{j\in[\![1,m-1]\!]}$ is a vector of independent Bernoulli variables, with parameter π. For $j \in [\![1, m-1]\!]$, Z_j indicates whether the comparison with the category j is correct ($Z_j = 1$) or not ($Z_j = 0$). C is the vector containing the $m-1$ results of the comparisons depending on both Z and the parameter μ. It is defined as follows:

$$\forall j \in [\![1, m-1]\!], C[j] = \begin{cases} (j < \mu) & \text{if } Z_j = 1 \\ (j \not< \mu) & \text{if } Z_j = 0 \end{cases}$$

The GOD model will generate $x \in [\![1, m]\!]$ such that $x \sim \mathcal{U}(\text{argmin}_{k\in[\![1,m]\!]} ||c - E_k||_1)$. We can interpret this as a probability maximization as stated in Theorem 3. The graphical model associated with this probabilistic model is depicted on Figure 2.

Definition 3 (Heaviside vector)**.** *For $k \in [\![1, m]\!]$, we define:*

$$E_k := (1)^{k-1}(0)^{m-k} = (\underbrace{1, \dots, 1}_{k-1}, \underbrace{0, \dots, 0}_{m-k}).$$

Theorem 3. *If we suppose that the prior distribution of μ is uniform over $[\![1, m]\!]$ and $\pi > \frac{1}{2}$, then $\forall c \in \{0,1\}^{m-1}$,*

$$\underset{k\in[\![1,m]\!]}{\text{argmax}} \Pr(\mu = k | C = c) = \underset{k\in[\![1,m]\!]}{\text{argmin}} ||c - E_k||_1 .$$

The proof can be found in appendix 8.

2.4.2 Parameter estimation

We want to estimate π and μ from a sample $(1, \dots, m)$ with weights $W \in \mathbb{R}_+^m$ generated by the GOD model. We aim at maximizing the likelihood of the sample : $\Pr(W|\pi, \mu)$. We proceed as explained in section 2.2.2.

Likelihood evaluation

Definition 4. *We define for $x \in [\![1, m]\!]$,:*

$$\mathcal{C}_x := \left\{ c \in \{0,1\}^{m-1} \,|x \in \underset{k\in[\![1,m]\!]}{\text{argmin}} ||c - E_k||_1 \right\}$$

Definition 5. *We define for $x \in [\![1, m]\!], \mu \in [\![1, m]\!], d \in [\![0, m-1]\!]$:*

$$u(\mu, x, d) := \sum_{c\in\mathcal{C}_x / ||c-E_\mu||_1 = d} \left| \underset{k\in[\![1,m]\!]}{\text{argmin}} ||c - E_k||_1 \right|^{-1}$$

Using these definitions we have the expression of the likelihood of a single observation.

Theorem 4 (Observation likelihood)**.**

$$\Pr(x|\pi, \mu) = \pi^{m-1} \sum_{d=0}^{m-1} \left(\frac{1-\pi}{\pi} \right)^d u(\mu, x, d)$$

Proof. See appendix 9. □

We can notice that once u is computed, the likelihood of a single observation can be computed in $\Theta(m)$ operations and the weighted likelihood can be computed in $\Theta(m^2)$ operations.

Computing the coefficients u The computation of the coefficients u can be done in $\mathcal{O}(m^2 2^m)$ time (see appendix E.4 for more details). We believe that it might be possible to compute it in polynomial time, with more research. Although still costly, it only needs to be computed once for a given m and can be stored in $\mathcal{O}(m^3)$ space.

Log-concavity

Conjecture 2. $\forall \mu \in [\![1, m]\!], \forall x \in [\![1, m]\!]$,

$$\pi \mapsto \Pr(x|\pi, \mu)$$

is log-*concave on* $[\frac{1}{2}, 1]$.

Like for the BOS model, we did not manage to prove this conjecture for all m. Moreoever, for numerical reasons, we did not manage to apply the same method as for the BOS model to verify it. As explained in section 2.2.2, we directly have that $\forall \mu \in [\![1, m]\!], \pi \mapsto L_W(\pi, \mu)$ is concave. Hence we can use a ternary search algorithm to estimate π for a given μ.

Conclusion As presented in section 2.2.2, we can estimate μ, π in $\mathcal{O}(m^3 \log \frac{1}{\varepsilon})$ operations once the coefficients u are computed. The coefficients u can be computed in $\mathcal{O}(m^2 2^m)$ operations and stored in $\mathcal{O}(m^3)$ space. This may seem costly, but m is usually small. (In comparison in Biernacki and Jacques [2016] they assume that $m \leqslant 7$)

3 Experiments

In this section, we try to evaluate the performance of the two models on different datasets. We first present the experimental setup and the datasets used for the experiments. We then present the results obtained for the estimation algorithms on synthetic data and on real-life datasets. Finally, we try to discuss the results obtained and the relevance of the models.

The goal of these experiments is to compare the two models but also to individually test their ability to cluster ordinal datasets and to check whether they are able to generalize to real-life datasets.

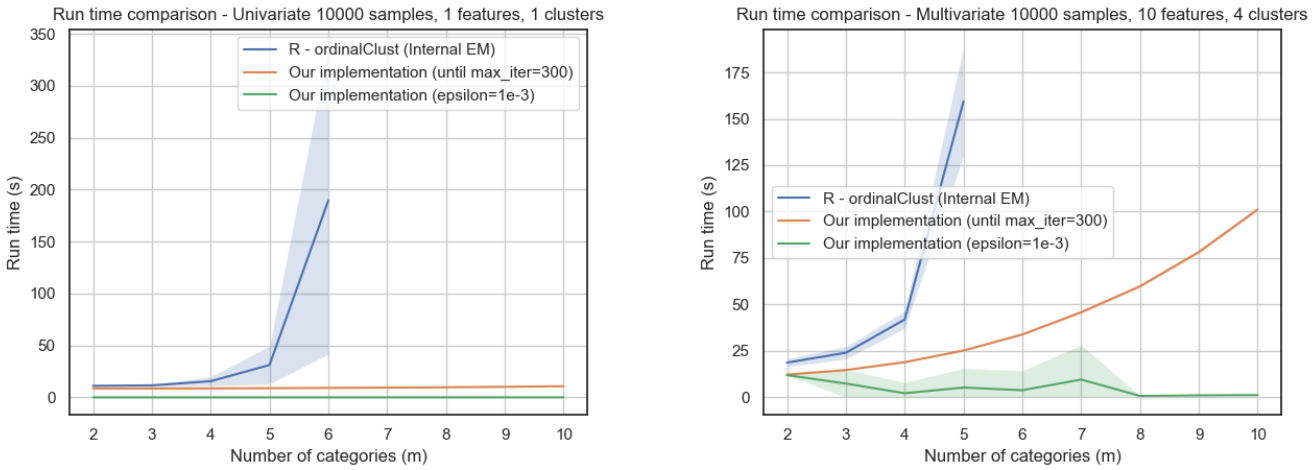
All the experiments and table are reproducible using the provided code and datasets and the fixed seeds.

3.1 Synthetic data

Experimental setup. In this section, we propose to test the AECM algorithm for the BOS and the GOD model on synthetic data in order to check the ability of our proposed estimation methods to correctly estimate the parameters of the dataand to cluster the datasets.

Runtimes. The runtimes of the AECM algorithm implementation with exponential complexity from Biernacki and Jacques [2016] are compared to our implementation of the AECM algorithm with polynomial complexity. Figure 3 compare both complexities for multiple runtimes for both univariate and multivariate AECM runs. The runtimes are reported with different number of categories in the dataset m. While the original implementation struggles to go further than $m = 6$, our implementation can easily reach higher number of categories. The implementation ordinalClust Selosse et al. [2021] is used for fair comparison and it is allowed to converge with a predefined ε parameter (internal to the authors package that we cannot modify). The worst case complexity of our implementation is plotted (with 300 iterations without allowing it to terminate when converged) is plotted to highlight

the time gains. The runtimes in the case where our implementation is stopped at convergence with an ε parameter, taken small enough for the comparison, are also plotted.



(a) Univariate case

(b) Multivariate case

Figure 3: Runtime comparison of the AECM algorithm for the BOS estimation on multivariate datasets as a function of the number of categories. For the ordinalClust package Selosse et al. [2021], nbSEM = 300, nbSEMburn = 200 and for our implementation, we run the model for 300 iterations in the first curve (worst case) and set $\varepsilon = 10^{-3}$ in the second curve. For every measurement, 10 datasets were generated and the average runtime is reported.

For all the experiments, the runtimes are measured on the same machine for all the tests and over 10 runs.

Evaluating the estimation error. We generate data from the BOS and the GOD model with different parameters and with both a random initialization of the parameters and an initialization of the parameters using the K-Means algorithm and then run the AECM algorithm on the generated data to estimate the parameters. We then compare the estimated parameters with the true parameters using the L_1 distance between the two vectors similarly to Biernacki and Jacques [2016]. We repeat this process multiple times for different parameters and average the results to obtain the metrics presented in Table 2 and Table 1 in Appendix A. Runtimes are also measured on the same machine for all the algorithms to evaluate their efficiency. The results show that on average, the parameter estimation is better with a random initialization than with a KMeans initialization. This is likely due to the fact that the KMeans initialization is not adapted to the ordinal nature of the data, and converges to local minima that are further from the ones from the initial distribution. Both the BOS and the GOD model parameters are being estimated with comparable accuracy.

Clustering performance. We also generate data with multiple clusters from different distributions and then run the AECM algorithm on the generated data to estimate the clusters. The goal of this experiment is to check the ability of the models to correctly cluster the data and to check whether the models are able to generalize to different distributions. The distributions used are the BOS model, the GOD model and discretized blobs. The following clustering algorithms are used for comparison: K-Means, Gaussian Mixture Models and the BOS and GOD models. The ARI score (section 3.2.2) is used to measure the clustering performance. The results are presented in Table 3 of Appendix A. We notice that the data generated from specialized ordinal models like BOS or GOD are better

classified by the BOS and GOD models than by the other models. At the same time, the BOS and GOD models are worse than KMeans and GMM on the blobs dataset. This is not surprising since the BOS and GOD models are specifically designed for ordinal data, and it highlights the importance of using the right model for the right data.

Visualizing the clusters. In order to get a better idea of the differences between the clustering methods, t-SNE visualizations [Van der Maaten and Hinton, 2008] are plotted for different distributions and clustering methods. The visualizations project the categorical data points in a continuous space and allow checking whether the estimated clusters are coherent with the true clusters. Since categorical data is used, it is more difficult to separate the clusters with smaller dimensional data and when the number of categories is small. Multiple datasets are generated with different parameters in order to highlight these differences. Moreover, as seen in the previous paragraph, it is also easier to cluster the data when the number of categories or features are high. The results are presented in Figure 4 of Appendix A. While the KMeans algorithm identifies clusters that are well separated visually, it fails to identify the true clusters in the case of the BOS and GOD models, especially when the number of categories is small. Moreover, even in the case where even 4 features are used, we also notice that categorical clusters are not easy to separate as shown by the obtained ARI scores. This highlights the difficulty of clustering categorical data even with specific algorithms designed to do so for small number of features or small number of categories.

3.2 Real-life datasets

3.2.1 Datasets

One of the main goal of the experiments is to test the ability of the models to generalize to real-life datasets. We therefore propose to test the illustrated methods on real world datasets to check the usefulness of the models on different real-life situations. Since the algorithm is specifically designed for ordinal observations, the datasets need to be adapted for the task. One way to apply to obtain real-life datasets is to quantize continuous datasets of observations that can be categorized (e.g. movies, store products, species...) [Skubacz and Hollmén, 2000]. Another interesting approach could be to test the models on tasks that they were not specifically designed for. This could allow seeing how they can generalize and whether they are applicable to a broader class of problems. We therefore propose to test the ability to cluster observations of binary features into different animal species.

Zoo Dataset. The zoo dataset consists of multiple features describing 101 different animals, with most of them being binary variables associated to a characteristic of the animal (hair, feathers, eggs, milk, ...) [Forsyth, 1990]. Every animal belongs to one of 6 classes.

Car Evaluation Dataset. The car evaluation dataset consists of multiple features describing 1728 different cars, with most of them being ordinal variables associated to a characteristic of the car (buying price, maintenance price, number of doors, ...) [Bohanec, 1997]. Every car belongs to one of 4 classes.

Hayes-Roth Dataset. The Hayes-Roth dataset consists of multiple features describing 132 different persons, with most of them being binary variables associated to a characteristic of the person (has a PhD, is married, is a professional, ...) [Hayes-Roth and Hayes-Roth, 1989]. Every person belongs to one of 3 classes.

Caesarian Dataset. The Caesarian dataset is a dataset describing 80 different patients with multiple features associated to the patient (age, delivery number, delivery time, blood pressure, ...) [Amin and Ali, 2018]. Every patient belongs to one of 2 classes.

The advantage of these datasets is that they are small enough to be able to compute the exact likelihood of the data given the model and the parameters. This allows to check whether the models are able to correctly fit the data.

Nursery School Dataset. The Nursery School dataset is a dataset describing 12960 different children with multiple features associated to the child (parents' occupation, family status, social conditions, ...) [Rajkovic, 1997]. Every child belongs to one of 5 classes. These classes are can also be interpreted as ordinal and represent the subjective quality of the nursery school that they attend.

3.2.2 Evaluation method.

For most of the real-life evaluation datasets, we will use classification tasks to check the ability to cluster with respect to pre-existing classes. This allows the evaluation framework to be easier to define. However, the results are very sensitive to the initial parameters used. In our evaluation, we keep the results for a unique seed, but it might be interesting to try different initialization scenarios in a real-life situation when trying to fit new data. Moreover, we are evaluating on the classification task, but the classes might not necessarily be the same as the clusters found (multiple classifications are possible in a dataset depending on the task).

In order to correctly associate the predicted clusters with the true clusters, we need to define a strategy that matches each predicted clusters with a true cluster number which will minimize a given criterion. In order to do so, we propose two methods:

- The first one consists in sorting the histograms of the predicted clusters and the true clusters and then matching the two sorted lists by assigning the predicted clusters to the true cluster in the same sorted order.

  This method is naive because it does not take into account the distribution of the real clusters according to the true labels for the matching.

- The second method consists in solving the Assignment Problem [Kuhn, 1955] with the cost matrix being the distance between the histograms of the predicted clusters and the true clusters. This method takes into account the distribution of the real clusters according to the true labels for the matching. We can easily solve it using any Optimal Transport algorithm (or by defining the Linear Programming problem and solving it using an LP solver).

Figure 5 of appendix B shows that the optimal matching when considering the assignment matrix is a better choice in the case of the Zoo dataset for example and that the classes in the predicted distribution are assigned to the correct true class with respect to their proportions.

The evaluation metrics used to compare the different models are the F1-score, and the Accuracy score in the cases where the datasets are suited for classification and the Wasserstein distance and the Adjusted Rand Index (ARI).

- The F1-score is the harmonic mean of the precision and the recall for classification problems.

- The Wasserstein distance is a measure of the distance between two probability distributions [Ramdas et al., 2017]. It measures the cost of transforming one distribution into the other using the optimal transport plan which in this case is the matching obtained as described above.

$$W(\hat{y}, y) = \min_{\gamma \in \Gamma(\hat{y}, y)} \sum_{i,j} \gamma_{i,j} \, ||i - j|| \,, \tag{4}$$

  where $\Gamma(\hat{y}, y)$ is the set of all possible matchings between the predicted clusters and the true clusters and $\gamma_{i,j}$ is the probability of matching the predicted cluster i with the true cluster j (i.e. it is the proportion of the samples in the predicted cluster i that are in the true cluster j) for the matching.

- The ARI is a measure of the similarity between two clusterings of the same dataset. It is a function that outputs a value between -0.5 and 1, where 1 means that the two clusterings are identical, 0 means that the two clusterings are independent (random) and -0.5 means that the two clusterings are as different as possible. The ARI is symmetric and therefore does not take into account the order of the clusters [Steinley, 2004].

$$\text{ARI}(\hat{y}, y) = \frac{\sum_{i,j} \binom{n_{i,j}}{2} - \left[\sum_i \binom{\hat{n}_i}{2} \sum_j \binom{n_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{\hat{n}_i}{2} + \sum_j \binom{n_j}{2}\right] - \left[\sum_i \binom{\hat{n}_i}{2} \sum_j \binom{n_j}{2}\right] / \binom{n}{2}}, \tag{5}$$

  where $n_{i,j}$ is the number of samples that are in the predicted cluster i and in the true cluster j, $\hat{n}_i$ is the number of samples in the predicted cluster i and n_j is the number of samples in the true cluster j.

3.2.3 Experiments with real-life datasets

To do so, we also use simple clustering algorithms to compare the performance of the BOS model on data that is adapted (ordinal) with algorithms that are not specifically designed for this kind of data such as K-Means [MacQueen et al., 1967] and Gaussian Mixture Models [Reynolds et al., 2009]. The results are presented in Table 4 in the appendix B. One interesting observation is that our implementation of the clustering methods scale well with the number of samples when compared to a Gaussian Mixture Model for example. During the experiments, the final results were very sensitive to the initial parameters used, most notably the proportion of each cluster, and initialization are likely to get stuck on a local minimum because of the nature of AECM. In order to get a better idea of the differences between the clustering methods, we also plot t-SNE visualizations [Van der Maaten and Hinton, 2008] for different datasets and the multiple models in Appendix B. The histogram and assignment matrix of the Zoo dataset are also provided in Appendix B in order to get a better understanding of the different assignments obtained in these settings for different models.

4 Conclusion

In this study, we examined model-based clustering for ordinal data, focusing on the Binary Ordinal Search (BOS) and an alternative we named the Globally Ordinal Distribution (GOD) models. We present an efficient algorithm for parameter estimation in these models, featuring a polynomial algorithm for the BOS model, an improvement over the exponential algorithm proposed in existing literature. Our objective was to comprehend and assess their effectiveness in clustering and classifying ordinal data, contrasting with more traditional methods such as K-Means and Gaussian Mixture Models. Our investigation encompassed both synthetic and real-world datasets, offering a comprehensive perspective on the models' performance across various scenarios.

The experiments on synthetic data confirmed the theoretical foundations of the BOS and GOD models. When applied to real-world datasets, the results were more nuanced. While both BOS and GOD models performed competitively in certain scenarios, they did not universally outperform the traditional methods. This suggests that while specialized ordinal models are interesting, especially in scenarios where the ordinal nature of data is pronounced, they are not a default solution. It is essential to consider the specific characteristics of the dataset and the computational resources available when choosing the appropriate clustering method.

Different visualizations also provide further insights into how the models partition the data space. They revealed that while the clusters identified by the BOS and GOD models often made intuitive sense, they sometimes differ significantly from those identified by K-Means and Gaussian Mixture Models. This highlights the different assumptions and approaches these models take when learning the structure within data.

In conclusion, the study reaffirms the potential of model-based clustering for ordinal data, particularly highlighting the BOS and GOD models as valuable tools. However, it also demonstrates that the choice of model should be informed by both the nature of the data and the practical constraints of the problem at hand. Further research could explore further refinements to these models, more extensive comparisons with other methods, and applications to a broader range of real-world scenarios. Moreover, it might also be interesting to look for an even more efficient way to compute the coefficients for the likelihood of the GOD model.

References

Alan Agresti. *Analysis of ordinal categorical data*, volume 656. John Wiley & Sons, 2010.

Muhammad Amin and Amir Ali. Caesarian Section Classification Dataset. UCI Machine Learning Repository, 2018. DOI: https://doi.org/10.24432/C5N59X.

Christophe Biernacki and Julien Jacques. Model-based clustering of multivariate ordinal data relying on a stochastic binary search algorithm. *Statistics and Computing*, 26:929–943, 2016.

Marko Bohanec. Car Evaluation. UCI Machine Learning Repository, 1997. DOI: https://doi.org/10.24432/C5JP48.

Angela D'Elia and Domenico Piccolo. A mixture model for preferences data analysis. *Computational Statistics & Data Analysis*, 49(3):917–934, 2005.

Richard Forsyth. Zoo. UCI Machine Learning Repository, 1990. DOI: https://doi.org/10.24432/C5R59V.

Barbara Hayes-Roth and Frederick Hayes-Roth. Hayes-Roth. UCI Machine Learning Repository, 1989. DOI: https://doi.org/10.24432/C5501T.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

Marica Manisera and Paola Zuccolotto. Modeling rating data with nonlinear cub models. *Computational Statistics & Data Analysis*, 78:100–118, 2014.

Xiao-Li Meng and David Van Dyk. The em algorithm—an old folk-song sung to a fast new tune. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 59(3):511–567, 1997.

Vladislav Rajkovic. Nursery. UCI Machine Learning Repository, 1997. DOI: https://doi.org/10.24432/C5P88W.

Aaditya Ramdas, Nicolás García Trillos, and Marco Cuturi. On wasserstein two-sample testing and related families of nonparametric tests. *Entropy*, 19(2):47, 2017.

Douglas A Reynolds et al. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663), 2009.

Margot Selosse, Julien Jacques, and Christophe Biernacki. ordinalclust: An r package to analyze ordinal data. *The R Journal*, 12(2), 2021.

Michal Skubacz and Jaakko Hollmén. Quantization of continuous input variables for binary classification. volume 1983, pages 42–47, 01 2000. ISBN 978-3-540-41450-6. doi: 10.1007/3-540-44491-2_7.

Douglas Steinley. Properties of the hubert-arable adjusted rand index. *Psychological methods*, 9(3): 386, 2004.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

A Synthetic data

AECM estimation for BOS and GOD distributions

Init.	n	$n_{clusters}$	d	n_{cats}	Runtime (s)	$\Delta\alpha$	$\Delta\mu$	$\Delta\pi$
kmeans	50	3	3	2	0.012	0.187	0.400	0.145
				3	0.048	0.155	0.378	0.073
			5	2	0.034	0.117	0.220	0.088
				3	0.079	0.111	0.240	0.046
		5	3	2	0.011	0.246	0.617	0.230
				3	0.079	0.218	0.667	0.133
			5	2	0.057	0.124	0.340	0.133
				3	0.103	0.129	0.427	0.088
	250	3	3	2	0.021	0.209	0.317	0.131
				3	0.057	0.148	0.400	0.067
			5	2	0.048	0.088	0.200	0.078
				3	0.086	0.091	0.213	0.040
		5	3	2	0.023	0.203	0.517	0.227
				3	0.087	0.209	0.522	0.130
			5	2	0.070	0.117	0.390	0.119
				3	0.136	0.090	0.393	0.074
random	50	3	3	2	0.023	0.155	0.367	0.109
				3	0.047	0.152	0.367	0.063
			5	2	0.038	0.073	0.130	0.057
				3	0.077	0.088	0.147	0.032
		5	3	2	0.035	0.149	0.533	0.157
				3	0.077	0.172	0.567	0.114
			5	2	0.061	0.106	0.410	0.117
				3	0.126	0.131	0.380	0.075
	250	3	3	2	0.031	0.152	0.283	0.104
				3	0.055	0.149	0.300	0.055
			5	2	0.045	0.072	0.140	0.052
				3	0.084	0.080	0.153	0.027
		5	3	2	0.044	0.144	0.517	0.148
				3	0.085	0.150	0.522	0.109
			5	2	0.069	0.119	0.390	0.105
				3	0.134	0.085	0.313	0.054

Table 1: Results of the experiments for the AECM algorithm on synthethic data with the GOD model. The parameters are the number of samples n, the number of clusters $n_{clusters}$, the dimension d and the number of categories n_{cats}. The deltas are the average of the L_1 distances between the true and estimated parameters after applying optimal transport to find the correct clusters. 10 different runs were made for each configuration and the average scores are reported for statistical significance.

Init.	n	$n_{clusters}$	d	n_{cats}	Runtime (s)	$\Delta\alpha$	$\Delta\mu$	$\Delta\pi$
kmeans	50	3	3	2	0.026	0.207	0.317	0.264
				3	0.051	0.216	0.356	0.146
			5	2	0.041	0.095	0.230	0.151
				3	0.143	0.112	0.233	0.099
		5	3	2	0.015	0.189	0.600	0.381
				3	0.111	0.250	0.744	0.250
			5	2	0.057	0.112	0.360	0.246
				3	0.188	0.095	0.367	0.144
	250	3	3	2	0.016	0.150	0.283	0.265
				3	0.089	0.146	0.344	0.139
			5	2	0.088	0.107	0.180	0.145
				3	0.112	0.090	0.213	0.072
		5	3	2	0.025	0.202	0.567	0.415
				3	0.150	0.238	0.611	0.241
			5	2	0.149	0.130	0.380	0.253
				3	0.385	0.126	0.427	0.140
random	50	3	3	2	0.026	0.122	0.200	0.161
				3	0.055	0.117	0.167	0.116
			5	2	0.053	0.056	0.130	0.094
				3	0.064	0.037	0.133	0.050
		5	3	2	0.061	0.118	0.283	0.234
				3	0.160	0.140	0.489	0.195
			5	2	0.127	0.087	0.260	0.192
				3	0.235	0.051	0.160	0.114
	250	3	3	2	0.084	0.143	0.150	0.082
				3	0.059	0.066	0.067	0.049
			5	2	0.085	0.049	0.040	0.052
				3	0.135	0.025	0.093	0.026
		5	3	2	0.074	0.109	0.150	0.153
				3	0.111	0.093	0.156	0.104
			5	2	0.207	0.060	0.120	0.114
				3	0.336	0.035	0.093	0.057

Table 2: Results of the experiments for the AECM algorithm no synthethic data with the BOS distribution. The parameters are the number of samples n, the number of clusters $n_{clusters}$, the dimension d and the number of categories n_{cats}. The deltas are the average of the L_1 distances between the true and estimated parameters after applying optimal transport to find the correct clusters. 10 different runs were made for each configuration and the average scores are reported for statistical significance.

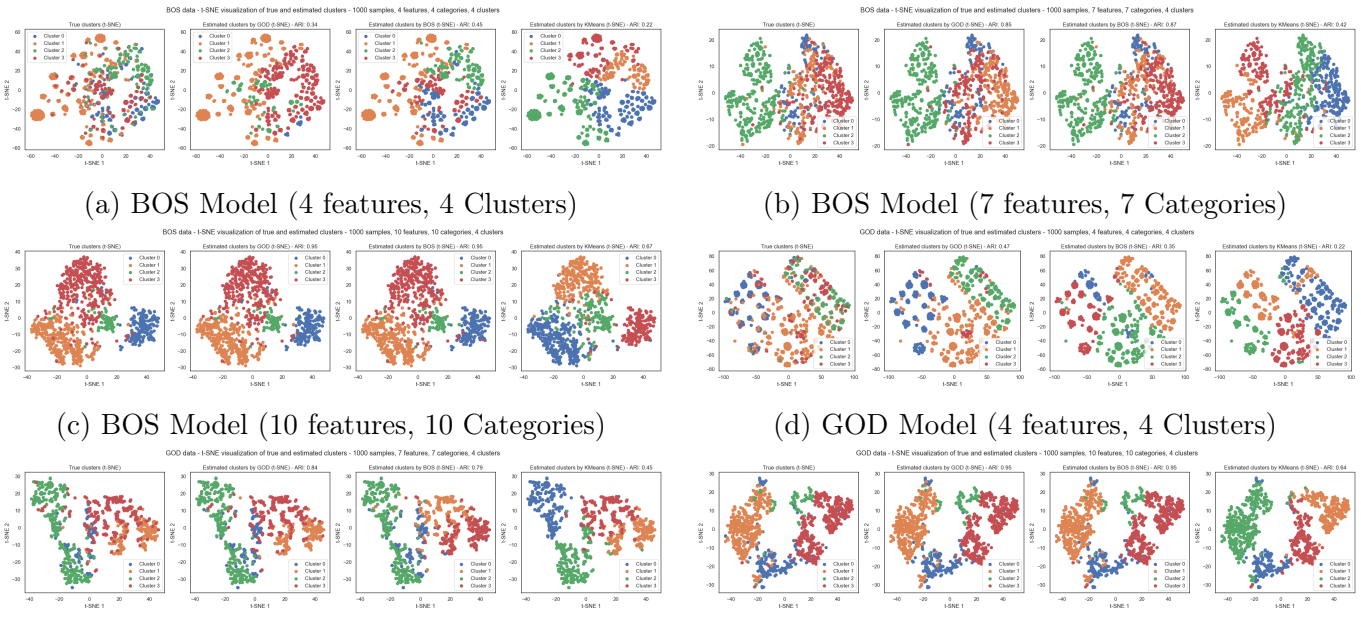
Synthetic data clustering

Data model	n	k	d	m	ARI BOS	ARI GOD	ARI KMeans	ARI GMM
BOS	10000	4	4	4	**0.389**	0.367	0.189	0.192
			7	7	**0.855**	0.840	0.395	0.450
			10	10	**0.972**	0.963	0.691	0.580
GOD	10000	4	4	4	0.260	**0.494**	0.308	0.308
			7	7	0.548	**0.828**	0.481	0.499
			10	10	0.707	**0.829**	0.649	0.614
Blobs	10000	4	4	4	0.747	0.928	**0.988**	**0.988**
			7	7	0.719	0.875	**0.993**	**0.993**
			10	10	0.976	0.246[a]	**1.000**	**1.000**

[a] likely stuck in a bad local minima

Table 3: Results of the clustering experiments on synthetic data. The metrics are the Adjusted Rand Index (ARI) for the different methods. The best results for each dataset and metric are highlighted in bold.

t-SNE plots for the synthetic data



(a) BOS Model (4 features, 4 Clusters)

(b) BOS Model (7 features, 7 Categories)

(c) BOS Model (10 features, 10 Categories)

(d) GOD Model (4 features, 4 Clusters)

(e) GOD Model (7 features, 7 Clusters)

(f) GOD Model (10 features, 10 Clusters)

Figure 4: Comparative t-SNE visualizations and assignment methods analysis. Subfigures (a) and (b) showcase BOS model distributions in 4D and 7D spaces, respectively, highlighting clusters and categories. Subfigure (c) visualizes the GOD model distribution in a 4D space. Subfigure (d) contrasts naive and optimal assignment methods post-clustering. Each visualization underscores the predictive capabilities and clustering accuracy across different dimensions and distributions.

B Real-life datasets

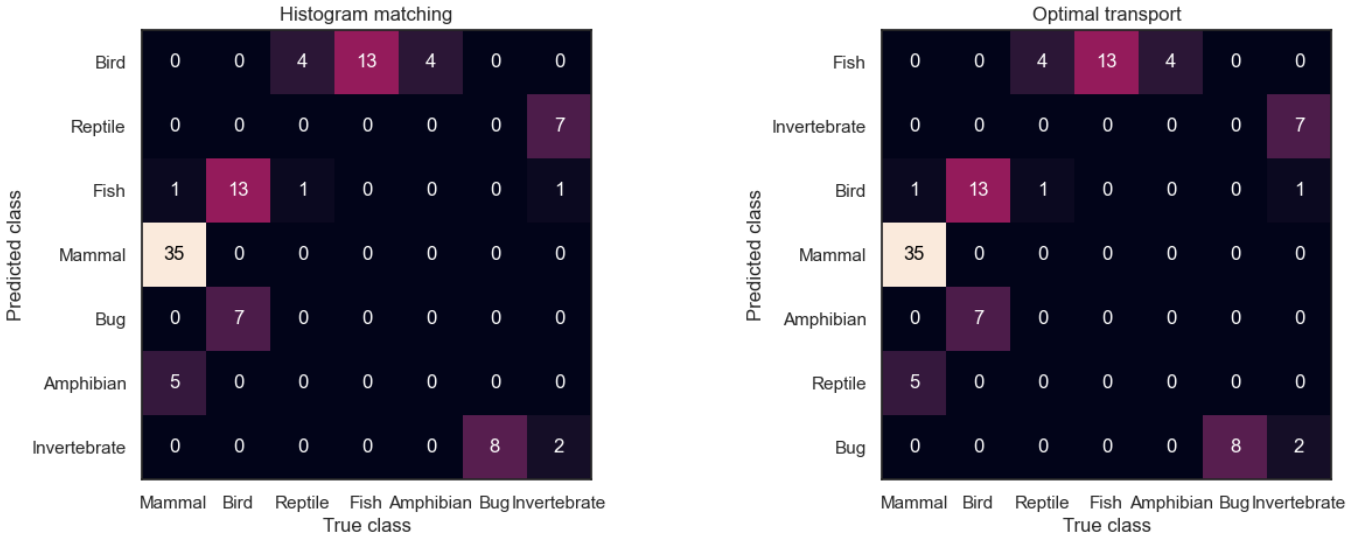


Figure 5: Illustration of the two assignment matrices from the different methods after clustering the Zoo dataset. On the left, the naive method and on the right, the optimal assignment method. The numbers in the matrices represent the number of samples in the predicted cluster i that are in the true cluster j.

t-SNE plots

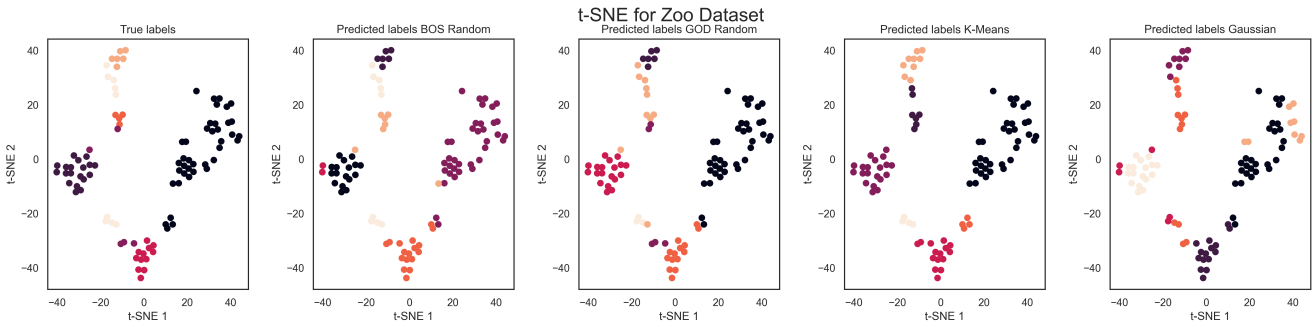


Figure 6: t-SNE visualization of the Zoo dataset with the true labels and the predicted clusters for the BOS model with random initialization, the GOD model with random initialization, the K-Means algorithm and the Gaussian Mixture Models.

Dataset	Method	Runtime (s)	F1	Accuracy	W_1	ARI
Zoo	**BOS Random**	0.93	0.89	0.89	0.20	0.85
	BOS K-Means	0.40	0.81	0.80	0.28	0.83
	GOD Random	0.63	**0.90**	**0.90**	**0.08**	**0.91**
	GOD K-Means	0.25	0.81	0.80	0.28	0.83
	K-Means	**0.01**	0.84	0.85	0.20	0.83
	Gaussian	0.52	0.77	0.74	0.55	0.66
Car Evaluation	**BOS Random**	0.20	0.46	0.38	**0.72**	0.01
	BOS K-Means	0.52	0.45	0.38	1.01	**0.04**
	GOD Random	0.18	**0.48**	**0.42**	0.74	-0.00
	GOD K-Means	0.60	0.39	0.31	1.03	-0.00
	K-Means	**0.01**	0.35	0.29	1.11	0.00
	Gaussian	0.04	0.41	0.33	1.24	0.01
Hayes-Roth	**BOS Random**	0.09	**0.45**	**0.46**	0.14	0.02
	BOS K-Means	0.09	0.34	0.33	0.16	-0.01
	GOD Random	0.26	0.40	0.41	0.34	0.01
	GOD K-Means	0.06	0.36	0.36	0.22	-0.01
	K-Means	**0.00**	0.34	0.33	0.16	-0.01
	Gaussian	0.03	**0.45**	0.45	**0.11**	**0.07**
Caesarian	**BOS Random**	0.26	**0.61**	**0.64**	0.21	**0.06**
	BOS K-Means	0.15	0.54	0.54	0.09	-0.01
	GOD Random	0.24	0.57	0.57	0.23	0.01
	GOD K-Means	0.12	0.56	0.56	0.06	0.00
	K-Means	**0.00**	0.56	0.56	0.09	0.00
	Gaussian	0.02	0.59	0.59	**0.04**	0.02
Nursery	**BOS Random**	**0.91**	**0.40**	**0.42**	0.84	0.04
	BOS K-Means	1.30	0.32	0.29	0.30	0.01
	GOD Random	5.31	0.40	0.39	**0.26**	0.03
	GOD K-Means	3.97	0.31	0.28	0.71	0.01
	K-Means	1.42	0.36	0.31	0.47	**0.07**
	Gaussian	33.32	0.34	0.29	0.54	0.05

Table 4: Results of the classification task for the different datasets and the proposed methods. The metrics are the F1-score, the accuracy, the Wasserstein distance and the adjusted rand index (ARI). The runtime is also reported. The best results for each dataset and metric are highlighted in bold italic and underlined.

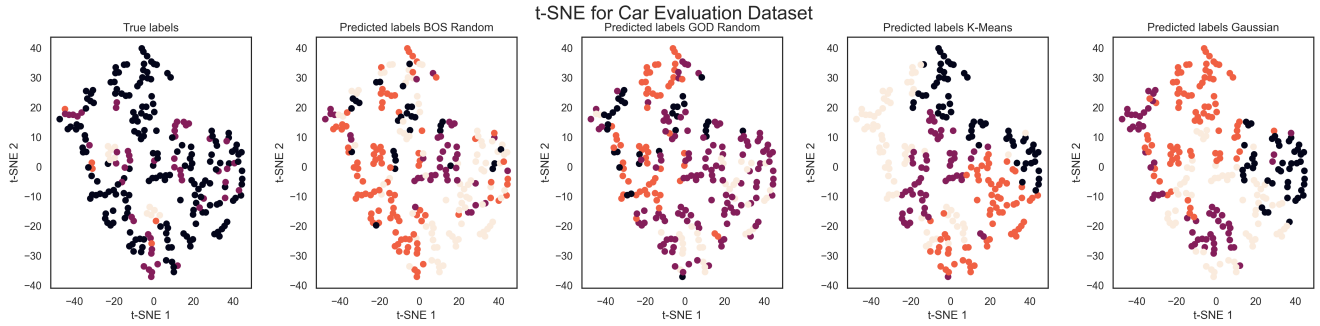


Figure 7: t-SNE visualization of the Car Evaluation dataset with the true labels and the predicted clusters for the BOS model with random initialization, the GOD model with random initialization, the K-Means algorithm and the Gaussian Mixture Models.

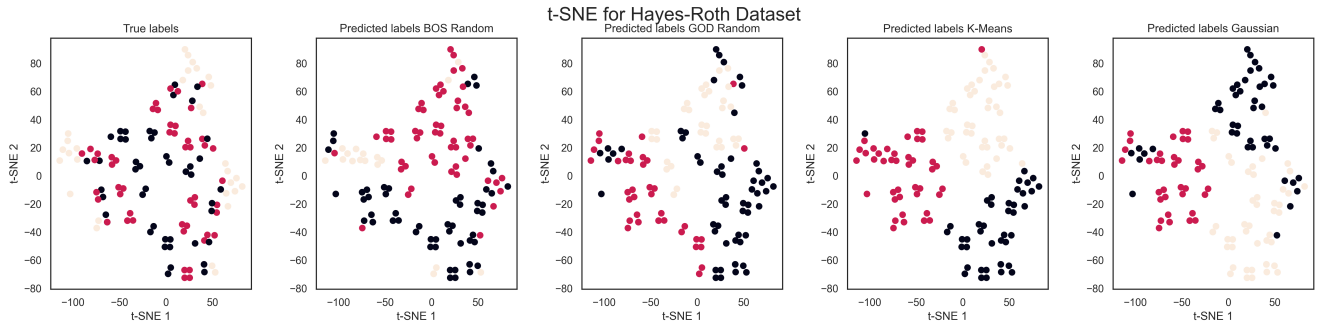


Figure 8: t-SNE visualization of the Hayes-Roth dataset with the true labels and the predicted clusters for the BOS model with random initialization, the GOD model with random initialization, the K-Means algorithm and the Gaussian Mixture Models.

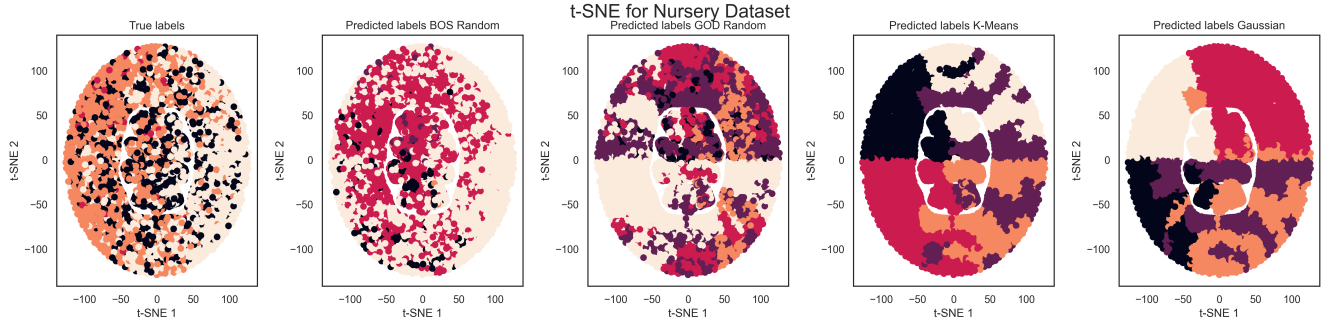


Figure 9: t-SNE visualization of the Nursery School dataset with the true labels and the predicted clusters for the BOS model with random initialization, the GOD model with random initialization, the K-Means algorithm and the Gaussian Mixture Models. While continuous clustering algorithms separate the circle as expected, categorical algorithms are a little more subtle and adapted.

Assignment matrix and histograms

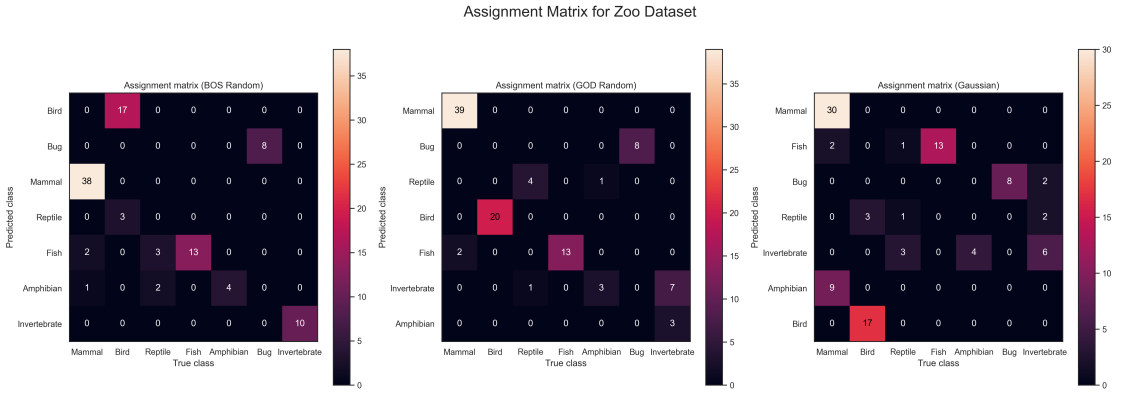


Figure 10: Assignment matrix for the Zoo dataset with different methods.

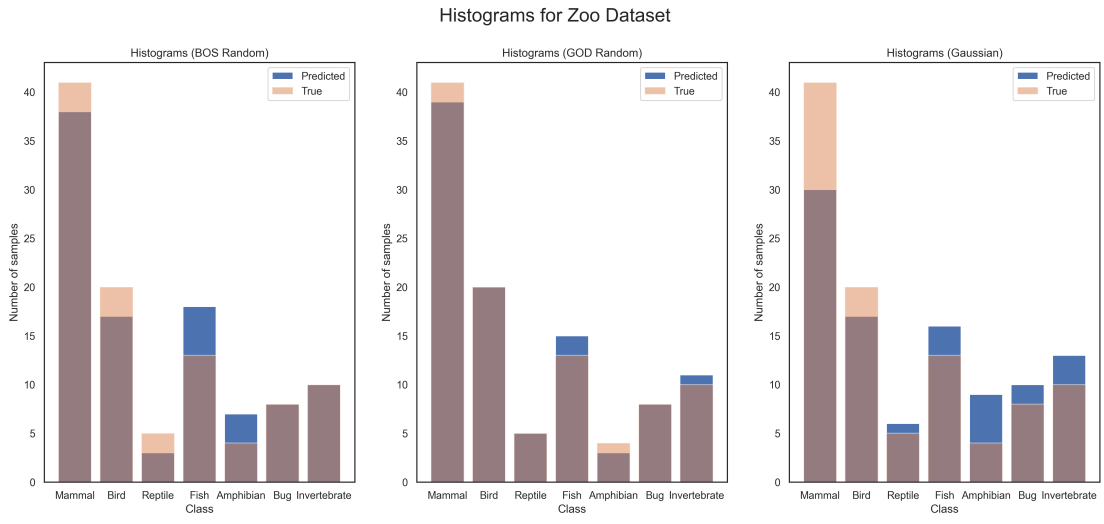


Figure 11: Histograms for the Zoo dataset with different methods.

C Generic proofs

C.1 Ternary search algorithm

Algorithm 1 Ternary search

Require: f concave on $[a, b]$, $\varepsilon > 0$
Ensure: For y the returned value, $\exists x \in \operatorname{argmax}_{[a,b]} f$, $|y - x| < \varepsilon$

```
function TernarySearch(f, a, b, ε)
    if b − a < ε then
        return (a+b)/2
    end if
    c ← a + (b−a)/3
    d ← a + 2(b−a)/3
    if f(c) ⩾ f(d) then
        return TernarySearch(f, a, d, ε)
    else
        return TernarySearch(f, c, b, ε)
    end if
end function
```

Theorem 5. *Let f be a concave function on $[A, B]$. The ternary search algorithm returns a value y such that $\exists x \in \operatorname{argmax}_{[A,B]} f$, $|y - x| < \varepsilon$.*

Proof. The algorithm stops when the length of the interval is less than ε. At each iteration, the length of the interval is multiplied by $\frac{2}{3}$. Hence the algorithm terminates.

We note $[a, b]$ the interval on which the algorithm is called and $[A, B]$ the initial interval. We prove the following invariant: at each iteration of the algorithm, the interval $[a, b]$ is such that $\exists x \in \operatorname{argmax}_{[a,b]} f, a \leqslant x \leqslant b$.

- At the beginning of the algorithm, we have $a = A$ and $b = B$. The invariant is true.
- We now suppose that $f(c) \geqslant f(d)$. We want to prove that the invariant is true for the interval $[a, d]$. We just have to prove that for any $g \in]d, b]$, $f(g) \leqslant f(c)$ (i.e. g is not an argmax or if so c is alos one). As $d \in [c, g]$, we have $\lambda \in]0, 1]$ such that $g = (1 - \lambda)c + \lambda d$. As f is concave, we have $f(d) \geqslant (1 - \lambda)f(c) + \lambda f(g)$. As $f(c) \geqslant f(d)$, we have $f(c) - (1 - \lambda)f(c) \geqslant \lambda f(g)$ which gives $f(c) \geqslant f(g)$. Hence the invariant is true for the interval $[a, d]$.

  We can do the same reasoning for the case $f(c) < f(d)$ and the interval $[c, b]$.

□

Complexity analysis Indeed after iteration k, the length of the interval is $\left(\frac{2}{3}\right)^k (b - a)$ [4]:

$$\left(\frac{2}{3}\right)^k (b - a) < \varepsilon \Leftrightarrow \left(\frac{2}{3}\right)^k < \frac{\varepsilon}{b - a} \tag{6}$$

$$\Leftrightarrow k(\lg 2 - \lg 3) < \lg \frac{\varepsilon}{b - a} \tag{7}$$

$$\Leftrightarrow k > \frac{1}{\lg 3 - 1} \lg \frac{b - a}{\varepsilon} \tag{8}$$

[4] $\lg = \log_2$

As each iteration of the algorithm require two evaluations of f, for $\varepsilon = 2^{-p}$ and $b - a \leqslant 1$. if we note k_e the number of evaluations of f required by the algorithm, we have:

$$k_e = \left\lceil \frac{2}{\lg 3 - 1} p \right\rceil \approx 2.4p \tag{9}$$

C.2 Concavity

Lemma 1 (Concavity of log composed functions)**.** *For $f : I \to \mathbb{R}_+^*$ be a twice-differentiable function, we have that:*

$$\ln \circ f \text{ is concave} \iff f'^2 - ff'' \geqslant 0$$

Proof. We have that:

$$(\ln \circ f)' = \frac{f'}{f} \tag{10}$$

$$(\ln \circ f)'' = \frac{f''f - f'^2}{f^2} \tag{11}$$

Therefore, $\ln \circ f$ is concave if and only if $f'^2 - ff'' \geqslant 0$. □

D BOS Model proofs

D.1 Notations

For the whole section we will consider that e is a subset of $[\![1, m]\!]$ and that supposing that we know e means that we look a the random process where the starting set of categories is e. We will also note $e^{-,y}$, $e^{=,y}$ and $e^{+,y}$ the sets of categories that are respectively less than, equal to and greater than y in e and f any next set of categories considered in the BOS process. For example, $\Pr(f|e)$ is the probability of having as next set of categories f knowing that the current set of categories is e (you could imagine that we have j such that $e_j = e$ and $e_{j+1} = f$).

Definition 6. *We define $Correct(\mu, e, y, f)$ as the indicator function that f is the correct subset to choose in case of a perfect comparison i.e. if $\mu \in f$ or by default the closest to μ.*

Definition 7. *We define $Next(e, y)$ as the set of intervals that can be chosen after a comparison at breakpoint y in the interval e i.e. $Next(e, y) = \{e^{-,y}, e^{=,y}, e^{+,y}\}$ with $e = [\![l, u - 1]\!]$ and $y \in [\![l, u - 1]\!]$: $e^{-,y} = [\![l, y - 1]\!]$, $e^{=,y} = \{y\}$ and $e^{+,y} = [\![y + 1, u - 1]\!]$.*

D.2 Polynomiality

Lemma 2 (Transition probability)**.** $\forall m \in \mathbb{N}^*, \forall x \in [\![1, m]\!], \forall \mu \in [\![1, m]\!], \pi \in [0, 1], \forall e \subset [\![1, m]\!], \forall f \subset e$*:*

$$\Pr(f|x \in e, e, \mu, \pi) = \frac{1}{|e|} \sum_{y \in e} \left[\left(Correct(\mu, e, y, f) - \frac{|f|}{|e|} \right) \pi + \frac{|f|}{|e|} \right] \mathbb{1}\{x \in Next(e, y)\}$$

Note that $\mathbb{1}\{x \in Next(e, y)\} = 1$ only for one value of y and 0 for all the others. Moreover $\pi \mapsto \Pr(f|x \in e, e, \mu, \pi)$ is an affine function.

Proof. We have that, by marginalization over the breakpoint y:

$$\Pr(f|x \in e, e, \mu, \pi) = \sum_{y \in e} \Pr(f, y|x \in e, e, \mu, \pi) \tag{12}$$

$$= \sum_{y \in e} \Pr(f|y, x \in e, e, \mu, \pi) \Pr(y|x \in e, e, \mu, \pi) \tag{13}$$

$$= \sum_{y \in e} \Pr(f|y, x \in e, e, \mu, \pi) \frac{1}{|e|} \tag{14}$$

$$= \frac{1}{|e|} \sum_{y \in e} \Pr(f|y, x \in e, e, \mu, \pi) \tag{15}$$

Then by marginalization over the accuracy indicator z:

$$\Pr(f|x \in e, e, \mu, \pi) = \frac{1}{|e|} \sum_{y \in e} \sum_{z \in \{0,1\}} \Pr(f|y, x \in e, e, \mu, \pi, z) \Pr(z|y, x \in e, e, \mu, \pi) \tag{16}$$

$$= \frac{1}{|e|} \sum_{y \in e} [\Pr(f|y, x \in e, e, \mu, \pi, z = 1)\pi + \Pr(f|y, x \in e, e, \mu, \pi, z = 0)(1 - \pi)] \tag{17}$$

$$= \frac{1}{|e|} \sum_{y \in e} \left[\text{Correct}\left(\mu, e, y, f\right) \pi + \frac{|f|}{|e|}(1 - \pi) \right] \mathbb{1}\left\{x \in \text{Next}\left(e, y\right)\right\} \tag{18}$$

$$= \frac{1}{|e|} \sum_{y \in e} \left[\left(\text{Correct}\left(\mu, e, y, f\right) - \frac{|f|}{|e|} \right) \pi + \frac{|f|}{|e|} \right] \mathbb{1}\left\{x \in \text{Next}\left(e, y\right)\right\} \tag{19}$$

□

Lemma 3. *$\forall m \in \mathbb{N}^*, \forall x \in [\![1, m]\!], \forall \mu \in [\![1, m]\!], \pi \in [0, 1], \forall e \subset [\![1, m]\!]$:*

$$\pi \mapsto \Pr(x|x \in e, e, \mu, \pi)$$

is a polynomial function of degree at most $|e| - 1$.

Proof. Let $m \in \mathbb{N}^*, x \in [\![1, m]\!], \mu \in [\![1, m]\!], \pi \in [0, 1]$.

We proceed by strong induction on $|e|$.

- Initialization: $|e| = 1$:

  $$\Pr(x|x \in e, e, \mu, \pi) = \mathbb{1}\left\{e = \{x\}\right\}$$

  which is a polynomial function of degree 0.

- Induction: Suppose the result holds for all $f \subset [\![1, m]\!]$ of size less or equal than $|e| - 1$ and let us prove it for $|e|$.

  We marginalize over the next interval f and we have that:

  $$\Pr(x|x \in e, e, \mu, \pi) = \sum_{f \subset e} \Pr(x, f|x \in e, e, \mu, \pi) \tag{20}$$

  $$= \sum_{f \subset e} \Pr(x|f, x \in e, e, \mu, \pi) \Pr(f|x \in e, e, \mu, \pi) \tag{21}$$

We can then notice that $\Pr(x|f, x \in e, e, \mu, \pi)$ is 0 if $x \notin f$ and that e does not intervene in the BOS process anymore. Hence we can replace $\Pr(x|f, x \in e, e, \mu, \pi)$ by $\Pr(x|x \in f, f, \mu, \pi)$ and sum only over $f \subset e$ such that $x \in f$:

$$\Pr(x|x \in e, e, \mu, \pi) = \sum_{f \subset e; x \in f} \Pr(x|x \in f, f, \mu, \pi) \Pr(f|x \in e, e, \mu, \pi) \tag{22}$$

As $\Pr(f|x \in e, e, \mu, \pi)$ is a polynomial function of degree at most 1 (see lemma 2) and $\Pr(x|f, x \in f, f, \mu, \pi)$ is a polynomial function of degree at most $|f| - 1 \leqslant |e| - 2$ by induction hypothesis, we have that $\Pr(x|x \in e, e, \mu, \pi)$ is a polynomial function of degree at most $|e| - 1$.

Hence the result holds for all e. □

Theorem 6 (Likelihood is polynomial)**.** $\forall m \in \mathbb{N}^*, \forall x \in [\![1, m]\!], \forall \mu \in [\![1, m]\!]$*,:*

$$\pi \mapsto \Pr(x|\mu, \pi)$$

is a polynomial function of degree at most $m - 1$*.*

Proof. Let $m \in \mathbb{N}^*$, $x \in [\![1, m]\!]$ and $\mu \in [\![1, m]\!]$.

First we can introduce redondant knowledge as we start necessarily with the full set of categories, we can add its value as known. We have that $\Pr(x|\mu, \pi) = \Pr(x|e_1, \mu, \pi)$. We also now that $x \in e_1$ therefore $\Pr(x|\mu, \pi) = \Pr(x|x \in e_1, e_1, \mu, \pi)$.

We can now use the previous lemma 3 to conclude that $\Pr(x|\mu, \pi)$ is a polynomial function of degree at most $m - 1$. □

D.3 Concavity

Conjecture 3 (Log concavity of the BOS model)**.** $\forall m \in \mathbb{N}^*, \forall x \in [\![1, m]\!], \forall \mu \in [\![1, m]\!]$*:*

$$\pi \mapsto \Pr(x|\mu, \pi)$$

is log*-concave on* $[0, 1]$*.*

Unfortunately, we have not yet been able to prove this theorem. However, we have successfully verified with high confidence that this property holds for $m \leqslant 94$. Our methodological approach for this empirical result involves checking the negativity of the second derivative of the log-likelihood. To do so we compute the second derivative and consider only the numerator since the denominator is always positive. Since the numerator is a polynomial, we can apply the following tests to verify its negativity on $[0, 1]$:

- If all the coefficients of the polynomial are negative, then the polynomial is negative on $[0, 1]$.
- Otherwise, we check if Lemma 4 applies.
- If it does not, we perform an efficient verification of the negativity of the polynomial by evaluating it on multiple points in $[0, 1]$ as described below.

If any of these tests is successful, we can conclude that the second derivative is negative on $[0, 1]$ and therefore the log-likelihood is concave on $[0, 1]$.

Lemma 4. *Let P be a polynomial function of degree n on $[0,1]$. Let $(a_i)_{i=0}^n$ be the coefficients of P such that $P(x) = \sum_{i=0}^n a_i x^i$. Let $(p_i)_{i=0}^n$ be the positive part of $(a_i)_{i=0}^n$, i.e. $\forall i, p_i = \max(0, a_i)$. In that case, if $\sum_{i=1}^n p_i \leqslant -a_0$, then P is negative on $[0,1]$.*

Proof. Let $x \in [0,1]$, $P(x) = \sum_{i=0}^n a_i x^i$. Define for all i, $p_i = \max(0, a_i)$ and $n_i = \min(0, a_i)$.

We have that:

$$\sum_{i=1}^n p_i \leqslant -a_0 \implies \sum_{i=1}^n p_i x_i \leqslant -a_0 \qquad \text{since } x \in [0,1] \tag{23}$$

$$\implies \sum_{i=1}^n p_i x_i + \sum_{i=1}^n n_i x_i \leqslant -a_0 \tag{24}$$

$$\iff \sum_{i=1}^n a_i x_i \leqslant -a_0 \tag{25}$$

$$\iff P(x) \leqslant 0 \tag{26}$$

□

Efficient verification of the negativity on $[0,1]$ of a polynomial One straightforward approach is to evaluate the polynomial at a large number of points in the interval $[0,1]$ and check if it is always negative. However, this method lacks firm guarantees on the negativity of the polynomial and requires increasing computational resources as the number of points increases.

To address this, we propose a more efficient method for verifying the negativity of a polynomial on $[0,1]$ with confidence limited only by numerical errors of the machine.

Suppose we know an upper bound M on the derivative of the function on $[0,1]$. Then, for any $x \in [0,1]$ and $h \in \mathbb{R}_+$ such that $x + h \leqslant 1$, we have $P(x+h) \leqslant P(x) + hM$. Therefore, if $P(x) \leqslant 0$ and $h \leqslant -\frac{P(x)}{M}$, then $P(x+h) \leqslant 0$.

This result allows us to devise an adaptive algorithm for verifying the negativity of a polynomial on $[0,1]$ by sampling the polynomial at intervals of varying lengths. The upper bound M can be estimated by computing the sum of the positive coefficients of degree $i \geqslant 1$ of the derivative of the polynomial and the coefficient of degree 0 (regardless of sign). The algorithm is outlined in Algorithm 2.

Algorithm 2 Check polynomial negativity

Require: P a polynomial function represented by its coefficients $(a_i)_{i=0}^n$

```
Let M = ComputeDerivativeUpperBound(P')
Let x = 0
Let y = P(x)
while x < 1 and y ⩽ 0 do
    y = P(x)
    x = x + -y/M
end while
return y ⩽ 0
```

D.4 Efficient computation of the likelihood

D.4.1 Mathematical proofs

Lemma 5 (Symetries the likelihood)**.** $\forall m \in \mathbb{N}^*, \forall x \in [\![1, m]\!], \forall \mu \in [\![1, m]\!], \pi \in [0,1], \forall e = [\![l, u]\!] \subset [\![1, m]\!]$*:*

$$\Pr(x|x \in [\![l,u]\!], e = [\![l,u]\!], \mu, \pi) = \Pr(x-l|x-l \in [\![0,u-l]\!], e = [\![0,u-l]\!], \max(0,\mu-l), \pi)$$

Proof. It is a simple translation of the categories. The only tricks is to notice that if μ is stricly less than the interval or equal to the lower bound, it will not affect the correctness of any sub-interval. □

Definition 8. *As justified by the lemma 5, we can define the following notation:*

$$l(x,\mu,h) := \Pr(x|x \in [\![0,h-1]\!], e = [\![0,h-1]\!], \mu, \pi)$$

Theorem 7 (Computing the likelihood)**.** $\forall m \in \mathbb{N}^*, \forall x \in [\![1,m]\!], \forall \mu \in [\![1,m]\!], \forall \pi \in [0,1]$*:*

$$\begin{aligned} l(x,\mu,h) = &\frac{1}{h}\sum_{y=0}^{x-1} l(x,\mu,y)\left[\left(\mathbb{1}\{\mu < y\} - \frac{y}{h}\right)\pi + \frac{y}{h}\right] \\ &+ \frac{1}{h}\left[\left(\mathbb{1}\{\mu = x \vee (x=0 \wedge \mu \leqslant x) \vee (x = h-1 \wedge \mu \geqslant x)\} - 1\right)\pi + \frac{1}{h}\right] \\ &+ \frac{1}{h}\sum_{y=x+1}^{h-1} l(x-y, \max(0,\mu-y), h-y)\left[\left(\mathbb{1}\{\mu > y\} - \frac{h-y-1}{h}\right)\pi + \frac{h-y-1}{h}\right] \end{aligned} \tag{27}$$

Proof. First we marginalize over the breakpoint y:

$$\begin{aligned} l(x,\mu,h) &= \Pr(x|x \in [\![0,h-1]\!], e = [\![0,h-1]\!], \mu, \pi) && (28)\\ &= \sum_{y=0}^{h-1} \Pr(x, f = e^{-,y}|x \in [\![0,h-1]\!], e = [\![0,h-1]\!], \mu, \pi) \\ &+ \sum_{y=0}^{h-1} \Pr(x, f = e^{=,y}|x \in [\![0,h-1]\!], e = [\![0,h-1]\!], \mu, \pi) && (29)\\ &+ \sum_{y=0}^{h-1} \Pr(x, f = e^{+,y}|x \in [\![0,h-1]\!], e = [\![0,h-1]\!], \mu, \pi) \\ & && (30) \end{aligned}$$

Then we use the Bayes rule ($P(A,B) = P(A|B)P(B)$) to get likelihoods of x:

$$\begin{aligned} &= \sum_{y=0}^{h-1} \Pr(x|x \in [\![0,h-1]\!], f = e^{-,y}, \mu, \pi) \Pr(e^{-,y}|x \in [\![0,h-1]\!], e = [\![0,h-1]\!], \mu, \pi) \\ &+ \sum_{y=0}^{h-1} \Pr(x|x \in [\![0,h-1]\!], f = e^{=,y}, \mu, \pi) \Pr(e^{=,y}|x \in [\![0,h-1]\!], e = [\![0,h-1]\!], \mu, \pi) && (31)\\ &+ \sum_{y=0}^{h-1} \Pr(x|x \in [\![0,h-1]\!], f = e^{+,y}, \mu, \pi) \Pr(e^{+,y}|x \in [\![0,h-1]\!], e = [\![0,h-1]\!], \mu, \pi) \\ & && (32) \end{aligned}$$

Then we can get rid of the case where $x \notin f$ as it is 0:

$$
\begin{aligned}
&= \sum_{y=0}^{x-1} \Pr(x|x \in [\![0, y-1]\!], f = [\![0, y-1]\!], \mu, \pi) \Pr([\![0, y-1]\!] | x \in [\![0, h-1]\!], e = [\![0, h-1]\!], \mu, \pi) \\
&+ \quad \Pr(x|x \in \{x\}, f = \{x\}, \mu, \pi) \Pr(\{x\} | x \in [\![0, h-1]\!], e = [\![0, h-1]\!], \mu, \pi) \\
&+ \sum_{y=x+1}^{h-1} \Pr(x|x \in [\![y+1, h-1]\!], f = [\![y+1, h-1]\!], \mu, \pi) \\
&\quad \Pr([\![y+1, h-1]\!] | x \in [\![0, h-1]\!], e = [\![0, h-1]\!], \mu, \pi)
\end{aligned} \tag{33}
$$

(34)

We can apply the lemma 5 to the third term:

$$
\begin{aligned}
&= \sum_{y=x+1}^{h-1} \Pr(x|x \in [\![0, y-1]\!], f = [\![0, y-1]\!], \mu, \pi) \Pr([\![0, y-1]\!] | x \in [\![0, h-1]\!], e = [\![0, h-1]\!], \mu, \pi) \\
&+ \quad \Pr(x|x \in \{x\}, f = \{x\}, \mu, \pi) \Pr(\{x\} | x \in [\![0, h-1]\!], e = [\![0, h-1]\!], \mu, \pi) \\
&+ \sum_{y=0}^{x-1} \Pr(x-y-1|x-y-1 \in [\![0, h-1-y-1]\!], f = [\![0, h-1-y-1]\!], \max(0, \mu-y-1), \pi) \\
&\quad \Pr([\![y+1, h-1]\!] | x \in [\![0, h-1]\!], e = [\![0, h-1]\!], \mu, \pi)
\end{aligned} \tag{35}
$$

$$
\begin{aligned}
&= \sum_{y=x+1}^{h-1} l(x, \mu, y-1) \Pr([\![0, y-1]\!] | x \in [\![0, h-1]\!], e = [\![0, h-1]\!], \mu, \pi) \\
&+ \quad l(x, \mu, 1) \Pr(\{x\} | x \in [\![0, h-1]\!], e = [\![0, h-1]\!], \mu, \pi) \\
&+ \sum_{y=0}^{x-1} l(x-y-1, \max(0, \mu-y-1), h-y-1) \Pr([\![y+1, h-1]\!] | x \in [\![0, h-1]\!], e = [\![0, h-1]\!], \mu, \pi)
\end{aligned} \tag{36}
$$

(37)

First we have $l(x, \mu, 1) = 1$. Moreover, we can now use the lemma 2 to get replace the transition probabilities. In our case as we already selected the only possible interval for each breakpoint we have the only term of the sum where $\mathbb{1}\{x \in \text{Next}(e, y)\} = 1$ and the sum is reduced to a single term:

$$
\begin{aligned}
&= \sum_{y=x+1}^{h-1} l(x, \mu, y-1) \frac{1}{h} \left[\left(\text{Correct}(\mu, [\![0, h-1]\!], y-1, [\![0, y-1]\!]) - \frac{y}{h} \right) \pi + \frac{y}{h} \right] \\
&+ \quad \frac{1}{h} \left[\left(\text{Correct}(\mu, [\![0, h-1]\!], x, \{x\}) - \frac{1}{h} \right) \pi + \frac{1}{h} \right] \\
&+ \sum_{y=0}^{x-1} l(x-y, \max(0, \mu-y), h-y) \frac{1}{h} \\
&\quad \left[\left(\text{Correct}(\mu-y-1, [\![0, h-1-y-1]\!], h-y-1, [\![y, h-1]\!]) - \frac{h-y-1}{h} \right) \pi + \frac{h-y-1}{h} \right]
\end{aligned} \tag{38}
$$

(39)

We can then replace Correct $(\mu, [\![0, h-1]\!], \bullet, \bullet)$ by a logical expression. (We must take into account the special case of the first and last breakpoint):

$$= \frac{1}{h} \sum_{y=x+1}^{h-1} l(x, \mu, y) \left[\left(\mathbb{1}\{\mu < y\} - \frac{y}{h} \right) \pi + \frac{y}{h} \right]$$

$$+ \frac{1}{h} \left[\left(\mathbb{1}\{\mu = x \vee (x = 0 \wedge \mu \leqslant x) \vee (x = h-1 \wedge \mu \geqslant x)\} - 1 \right) \pi + \frac{1}{h} \right] \tag{40}$$

$$+ \frac{1}{h} \sum_{y=0}^{x-1} l(x-y, \max(0, \mu-y), h-y) \left[\left(\mathbb{1}\{\mu > y\} - \frac{h-y-1}{h} \right) \pi + \frac{h-y-1}{h} \right]$$

(41)

□

D.4.2 Algorithm

In the algorithm we consider that u contains polynomials and that $\times_P$ and $+_P$ are respectively the multiplication and addition of polynomials.

on the line 16, we use the a min to avoid computing the polynomial for μ outside the considered range. As if μ is greater than the interval or equal to the upper bound, the correctness of any sub-interval is not different.

Algorithm 3 BOS polynomial computation

Require: $m \in \mathbb{N}$
Ensure: $u[m-1, \mu-1, x-1] = \Pr(x|\mu, \pi)$ where we consider it as a polynomial equality

```
1: u ← array of size m × m × m initialised to 0
2: u[0] ← 1
3: for h ∈ [[2, m]] do
4:     for μ ∈ [[0, h − 1]] do
5:         for x ∈ [[0, h − 1]] do
6:             s = u[h − 1, μ, x]
7:             for y ∈ [[0, x − 1]] do
8:                 p ← u[h − y − 2, max(0, μ − y − 1), x − y − 1]
9:                 a ← (h−y−1)/h
10:                s ← s +_P p ×_P [(𝟙{μ > y} − a) π +_P a]
11:            end for
12:            c ← 𝟙{μ = x ∨ (x = 0 ∧ μ ⩽ x) ∨ (x = h − 1 ∧ μ ⩾ x)}
13:            a ← 1/h
14:            s ← s +_P (c − a)π +_P a
15:            for y ∈ [[x + 1, h − 1]] do
16:                p ← u[y − 1, min(μ, y − 1), x]
17:                a ← y/h
18:                s ← s +_P p ×_P [(𝟙{μ < y} − a) π +_P a]
19:            end for
20:            u[h, μ, x] = s/h
21:        end for
22:    end for
23: end for
```

E GOD Model proofs

E.1 Probabilistic model

Theorem 8. *Assuming that the prior distribution of μ is uniform over $[\![1, m]\!]$ and $\pi > \frac{1}{2}$, then $\forall c \in \{0,1\}^{m-1}$,*

$$\underset{k\in[\![1,m]\!]}{\operatorname{argmax}} \Pr(\mu = k | C = c) = \underset{k\in[\![1,m]\!]}{\operatorname{argmin}} ||c - E_k||_1$$

Proof.

Lemma 6.

$$\Pr(C[i] = c[i] | \mu < i) = (1 - c[i])\pi + c[i](1 - \pi)$$
$$\Pr(C[i] = c[i] | \mu \not< i) = c[i]\pi + (1 - c[i])(1 - \pi)$$

Proof.

$$\begin{aligned}
\Pr(C[i] = c[i] | \mu \not< i) &= \Pr(C[i] = c[i] | Z[i] = 1, \mu \not< i) \Pr(Z[i] = 1) && (42)\\
&\quad + \Pr(C[i] = c[i] | Z[i] = 0, \mu \not< i) \Pr(Z[i] = 0) && (43)\\
&= \mathbb{1}\{c[i] = 1\} \Pr(Z[i] = 1) && (44)\\
&\quad + \mathbb{1}\{c[i] = 0\} \Pr(Z[i] = 0) && (45)
\end{aligned}$$

Indeed conditional on $\mu \not< i$ and $Z[i] = 1$, $C[i] = 1$. Similarly, conditional on $\mu \not< i$ and $Z[i] = 0$, $C[i] = 0$. Hence:

$$\begin{aligned}
\Pr(C[i] = c[i] | \mu \not< i) &= c[i] \Pr(Z[i] = 1) + (1 - c[i]) \Pr(Z[i] = 0) && (46)\\
&= c[i]\pi + (1 - c[i])(1 - \pi) && (47)\\
& && (48)
\end{aligned}$$

$$\begin{aligned}
\Pr(C[i] = c[i] | \mu < i) &= \Pr(C[i] = c[i] | Z[i] = 1, \mu < i) \Pr(Z[i] = 1) && (49)\\
&\quad + \Pr(C[i] = c[i] | Z[i] = 0, \mu < i) \Pr(Z[i] = 0) && (50)\\
&= (1 - c[i]) \Pr(Z[i] = 1) + c[i] \Pr(Z[i] = 0) && (51)\\
&= (1 - c[i])\pi + c[i](1 - \pi) && (52)
\end{aligned}$$

□

Lemma 7. $\forall c \in \{0,1\}^m, \forall k \in [\![1, m]\!]$,

$$\Pr(C = c | \mu = k) = \pi^{m-1-||c-E_k||_1}(1 - \pi)^{||c-E_k||_1}$$

Proof. Let us compute $\Pr(C = c | \mu = i)$ for $i \in [\![1, m]\!]$ by noticing that $C[i]|\mu$ are conditionally independent

$$\begin{aligned}
\Pr(C = c | \mu = k) &= \prod_{i=1}^{m-1} \Pr(C[i] = c[i] | \mu = k) && (53)\\
&= \prod_{i=1}^{k} \Pr(C[i] = c[i] | \mu \not< i) \prod_{i=k+1}^{m-1} \Pr(C[i] = c[i] | \mu < i) && (54)\\
& && (55)
\end{aligned}$$

The last line come from the fact that $\Pr(C[i] = c[i])$ only depends on whether $\mu < i$ or not. Hence, if $i \leqslant k$, then $\Pr(C[i] = c[i]|\mu = k) = \Pr(C[i] = c[i]|\mu \not< i)$ and if $i > k$, then $\Pr(C[i] = c[i]|\mu = k) = \Pr(C[i] = c[i]|\mu < i)$. We now apply the previous lemma to finish the proof.

$$\Pr(C = c|\mu = k) = \prod_{i=1}^{k-1} \Pr(C[i] = c[i]|\mu \not< i) \prod_{i=k}^{m-1} \Pr(C[i] = c[i]|\mu < i) \tag{56}$$

$$= \prod_{i=1}^{k-1} [c[i]\pi + (1 - c[i])(1 - \pi)] \prod_{i=k}^{m-1} [(1 - c[i])\pi + c[i](1 - \pi)] \tag{57}$$

$$= \pi^{\sum_{i=1}^{k-1} c[i]} (1 - \pi)^{\sum_{i=1}^{k-1} (1-c[i])} \pi^{\sum_{i=k}^{m-1} (1-c[i])} (1 - \pi)^{\sum_{i=k}^{m-1} c[i]} \tag{58}$$

$$= \pi^{\sum_{i=1}^{k-1} c[i] + \sum_{i=k}^{m-1} (1-c[i])} (1 - \pi)^{\sum_{i=1}^{k-1} (1-c[i]) + \sum_{i=k}^{m-1} c[i]} \tag{59}$$

$$= \pi^{m-1-\left[\sum_{i=1}^{k-1} (1-c[i]) + \sum_{i=k}^{m-1} c[i]\right]} (1 - \pi)^{\sum_{i=1}^{k-1} (1-c[i]) + \sum_{i=k}^{m-1} c[i]} \tag{60}$$

$$= \pi^{m-1-||E_k - c||_1} (1 - \pi)^{||E_k - c||_1} \tag{61}$$

□

$$\Pr(\mu = k|C = c) = \frac{\Pr(C = c|\mu = k) \Pr(\mu = k)}{\Pr(C = c)} \tag{62}$$

$$= \frac{\Pr(C = c|\mu = k) \Pr(\mu = k)}{\sum_{i=1}^{m} \Pr(C = c|\mu = i) \Pr(\mu = i)} \tag{63}$$

As μ is uniformly distributed over $[\![1, m]\!]$, $\Pr(\mu = k) = \frac{1}{m}$

$$\Pr(\mu = k|C = c) = \frac{\Pr(C = c|\mu = k)}{\sum_{i=1}^{m} \Pr(C|\mu = i)} \tag{64}$$

using Lemma 7:

$$\Pr(\mu = k|C = c) = \frac{\pi^{m-1-||c-E_k||_1} (1 - \pi)^{||c-E_k||_1}}{\sum_{i=1}^{m} \pi^{m-1-||c-E_i||_1} (1 - \pi)^{||c-E_i||_1}} \tag{65}$$

$$\tag{66}$$

As $\pi > \frac{1}{2}$, we conclude that:

$$\underset{k \in [\![1,m]\!]}{\operatorname{argmax}} \Pr(\mu = k|C = c) = \underset{k \in [\![1,m]\!]}{\operatorname{argmin}} ||c - E_k||_1$$

□

E.2 Likelihood expression

We recall the definition of $\mathcal{C}_x$ and $u(x, \mu, d)$:

Definition 9. *We define for $x \in [\![1, m]\!]$,:*

$$\mathcal{C}_x := \left\{ c \in \{0, 1\}^{m-1} \,|x \in \underset{k \in [\![1,m]\!]}{\operatorname{argmin}} ||c - E_k||_1 \right\}$$

Definition 10. *We define for* $x \in [\![1, m]\!], \mu \in [\![1, m]\!], d \in [\![0, m-1]\!]$*:*

$$u(\mu, x, d) := \sum_{c \in \mathcal{C}_x / ||c - E_\mu||_1 = d} \left| \underset{k \in [\![1,m]\!]}{\operatorname{argmin}} ||c - E_k||_1 \right|^{-1}$$

Lemma 8.

$$\Pr(x, c|\pi, \mu) = \mathbb{1}_{\mathcal{C}_x}(c) \pi^{m-1} \frac{\left(\frac{1-\pi}{\pi}\right)^{||c - E_\mu||_1}}{\left|\operatorname{argmin}_{k \in [\![1,m]\!]} ||c - E_k||_1\right|}$$

Proof. Using Bayes' theorem, then Lemma 7 and the fact that μ is uniformly distributed over the set defined by the argmin, we have:

$$\begin{aligned}
\Pr(x, C = c|\pi, \mu) &= \Pr(x|c, \pi, \mu) \Pr(C = c|\pi, \mu) && (67)\\
&= \mathbb{1}_{\mathcal{C}_x}(c) \Pr(x|c \in \mathcal{C}_x, \pi, \mu) \Pr(c|\pi, \mu) && (68)\\
&= \mathbb{1}_{\mathcal{C}_x}(c) \frac{\pi^{m-1-||c-E_\mu||_1} (1-\pi)^{||c-E_\mu||_1}}{\left|\operatorname{argmin}_{k \in [\![1,m]\!]} ||c - E_k||_1\right|} && (69)\\
&= \mathbb{1}_{\mathcal{C}_x}(c) \pi^{m-1} \frac{\left(\frac{1-\pi}{\pi}\right)^{||c-E_\mu||_1}}{\left|\operatorname{argmin}_{k \in [\![1,m]\!]} ||c - E_k||_1\right|} && (70)
\end{aligned}$$

□

Theorem 9 (Observation likelihood)**.**

$$\Pr(x|\pi, \mu) = \pi^{m-1} \sum_{d=0}^{m-1} \left(\frac{1-\pi}{\pi}\right)^d u(\mu, x, d)$$

Proof. By marginalizing over c and then using the previous lemma, we have:

$$\begin{aligned}
\Pr(x|\pi, \mu) &= \sum_{c \in \{0,1\}^{m-1}} \Pr(x, c|\pi, \mu) && (71)\\
&= \pi^{m-1} \sum_{c \in \mathcal{C}_x} \left(\frac{1-\pi}{\pi}\right)^{||c-E_\mu||_1} \left| \underset{k \in [\![1,m]\!]}{\operatorname{argmin}} ||c - E_k||_1 \right|^{-1} && (72)\\
&= \pi^{m-1} \sum_{d=0}^{m-1} \left(\frac{1-\pi}{\pi}\right)^d \sum_{c \in \mathcal{C}_x / ||c-E_\mu||_1 = d} \left| \underset{k \in [\![1,m]\!]}{\operatorname{argmin}} ||c - E_k||_1 \right|^{-1} && (73)
\end{aligned}$$

□

E.3 Concavity

Conjecture 4. $\forall \mu \in [\![1, m]\!], \forall x \in [\![1, m]\!]$

$$\pi \mapsto \Pr(x|\pi, \mu)$$

is log*-concave on* $\left[\frac{1}{2}, 1\right]$*.*

We were not able to prove this conjecture. However, as an indication of concavity, we have been able to verify numerically up to $m = 26$ that all the functions are increasing, then constant and then decreasing (with possibly an empty increasing or decreasing part) on $\left[\frac{1}{2}, 1\right]$.

In addition, we confirmed in the experiments that the ternary search algorithm worked well for the estimation of parameters.

E.4 Algorithm

We present here our algorithm to compute the coefficients u of the polynomial expansion of the likelihood of the GOD model.

Algorithm 4 GOD polynomial computation

Require: $m \in \mathbb{N}$

Ensure: u the coefficients of the polynomial expansion of the likelihood of the GOD model

$D : D[i,k] = ||bin(i) - E_k||\,, \forall i \in [\![0, 2^{m-1}-1]\!]\,, \forall k \in [\![1,m]\!]$

$is\_min : is\_min[i,k] = (||bin(i) - E_k|| = \min_{l \in [\![1,m]\!]} ||bin(i) - E_l||), \forall i \in [\![0, 2^{m-1}-1]\!]\,, \forall k \in [\![1,m]\!]$

$card\_min : card\_min[i] = |\{k \in [\![1,m]\!] \,|is\_min[i,k] = 1\}|\,, \forall i \in [\![0, 2^{m-1}-1]\!]$

$u : u[\mu, x, d] = 0, \forall \mu \in [\![1,m]\!]\,, \forall x \in [\![1,m]\!]\,, \forall d \in [\![0, m-1]\!]$

for $i \in [\![0, 2^{m-1}-1]\!]$ **do**
  for $x \in [\![1,m]\!]$ **do**
    if $is\_min[i, x-1]$ **then**
      for $\mu \in [\![1,m]\!]$ **do**
        $d = D[i, \mu - 1]$
        $u[\mu - 1, x - 1, d] += \frac{1}{card\_min[i]}$
      end for
    end if
  end for
end for

The algorithm quite simple. We first compute the distance between each possible binary vector in D. Then we use it to determine if an E_k is the closest to a binary vector and we store this in is_min. Finally, we use this information to compute the number of closest E_k to a binary vector in $card\_min$.

The computation can be done in $\mathcal{O}(m2^m)$ operations by noticing that it is possible to compute the distance between a binary vector and all the E_k in $\mathcal{O}(m)$ operations using the fact that the change of distance between two E_k is at most 1. Then computing is_min and $card\_min$ can be done in $\mathcal{O}(m2^m)$ operations.

Then we just need to compute the sum. Note that naively this would require $\mathcal{O}(m^2 2^m)$ operations. However, we can notice that for a given i and x, we only loop over μ if E_μ is the closest to i. On average, this seems to happen less than 2 times. Hence the complexity of the algorithm is $\mathcal{O}(m^2 2^m)$ without assuming this property and $\mathcal{O}(m2^m)$ if we assume it.

Mathematically this property is translated by the following conjecture:

Conjecture 5.

$$\sum_{c \in \{0,1\}^{m-1}} \left| \operatorname*{argmin}_{k \in [\![1,m]\!]} ||c - E_k||_1 \right| = 2^m - \binom{m}{\lfloor \frac{m}{2} \rfloor}$$