Clustering Multivariate Ordinal Data

Thomas Michel, Théo Rudkiewicz and Ali Ramlaoui

January 15, 2024

1 Introduction

The exploration of hidden structures within datasets is a crucial task for data scientists, and clustering serves as a valuable tool in this endeavor. 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 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 explores the less-explored domain of model-based clustering for ordinal data, specifically focusing on ordinal data derived from ordered categories. 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 it is assessed by specialists. The use of ordinal data enriches the comprehension of subjective opinions, behaviors, and hierarchical relationships across diverse research contexts. Over the years, various approaches have been proposed to define probability distributions for ordinal data, including modeling cumulative probabilities, constraining multinomial models to reflect ordinality, assuming ordinal data as discretization of continuous latent variables, and constructing distributions to meet specific properties.

Among these approaches, the work by **?**, studied in the context of this project, delves into an original strategy—modeling the hypothetical data generating process for ordinal data. While this general principle has found success in ranking data scenarios, the distinction in the data generating process becomes apparent for ordinal data. In this context, a search algorithm, specifically the binary search algorithm, emerges as a fitting choice, respecting the ordinal nature of data through comparisons without necessitating links to nominal or continuous distributions.

The proposed model, parameterized with a position parameter (modal category) and a precision parameter, exhibits desirable properties such as a unique mode, probability distribution decrease on either side of the mode, and the flexibility to accommodate uniform or Dirac distributions. Maximum likelihood estimation using an EM algorithm is employed, leveraging the binary search algorithm's latent variable interpretation. 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.

In this project, we aim to replicate and build upon the findings of **?**. We re-implemented their suggested probabilistic model, parameter estimation method, and model-based clustering algorithm in Python. Drawing inspiration from their approach, we propose an alternative probabilistic model with similar properties. The goal is to address computational limitations, enabling the clustering of more extensive datasets with potentially more categories than the previous method allows. We also present a preliminary analysis of this new method to justify the decreased computational cost of estimating parameters for this model. Additionally, we test **?**'s approach on real-world datasets and compare it to the proposed approach, along with baseline models. This is done on different datasets of multiple nature in order to check whether the proposed methods are successful in these settings in practice and what their advantages are. The ultimate goal is to check whether the gains are significant against methods that are not adapted for ordinal datasets, in order to decide whether these approaches are interesting to use in general as a default method of choice for this type of variable.

2 Method

The approach taken in this report follows the one proposed by **?**. It involves defining a probabilistic model based on the observation of categorical data. More precisely, the authors introduce a Stochastic Binary Search process that leads to the selection of a category.

For this particular model, we need to make the assumption that:

- The categories must be well-ordered (ordinal data): The categories are linearly ordered, and each non-empty subset contains the 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.

We will first define the binary ordinal search (BOS) model in the univariate case. The multivariate variant can be deduced by applying a similar search to each of the features independently. Then, we will discuss how to estimate the parameters of the model from a set of samples following the general method of the Expectation-Maximization algorithm (EM). Finally, we will show how the original authors extend EM to cluster ordinal data.

2.1 Probabilistic model

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.

The stochastic binary ordinal search unfolds as follows: Let m be the number of categories. For simplicity, we denote the categories using unique values from 1 to m. Then, for at most $m-1$ steps, we perform the following three operations. At step j, we start with a subset of all the categories, denoted as $e_j \subseteq \{1, ..., m\}$.

1. Sample a breakpoint y_j uniformly in e_j.
2. Draw an accuracy indicator z_j from a Bernoulli distribution with parameter π. 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^-, $e_j^=$, and e_j^+. 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.

For data with multiple ordinal features, the samples are generated independently with different parameters μ and π and possibly different numbers of categories for each feature.

Mixture model The BOS model can be extended to a mixture model by considering multiple BOS models with different parameters. In this case, the data is generated by first sampling a cluster k from a multinomial distribution with parameter α and then sampling the data from the BOS model with parameters μ_k and π_k.

We can also consider a multivariate version of the BOS model in addition to the mixture case where we have a BOS model for each feature independently. Each dimension is then concatenated to get the multivariate dataset that follows a multivariate BOS distribution.

Therefore, for p clusters, and d features, the parameters of the model are $\alpha \in \mathbb{R}^p$, $\mu \in \mathbb{R}^{p \times d}$ and $\pi \in \mathbb{R}^{p \times d}$ Since the features are independent, the probability of observing a sample x knowing it belongs to cluster k is given by:

$$\mathbb{P}(x|\alpha_k, \mu_k, \pi_k) = \prod_{j=1}^{d} \mathbb{P}(x_j|\alpha_k, \mu_{kj}, \pi_{kj}) \tag{1}$$

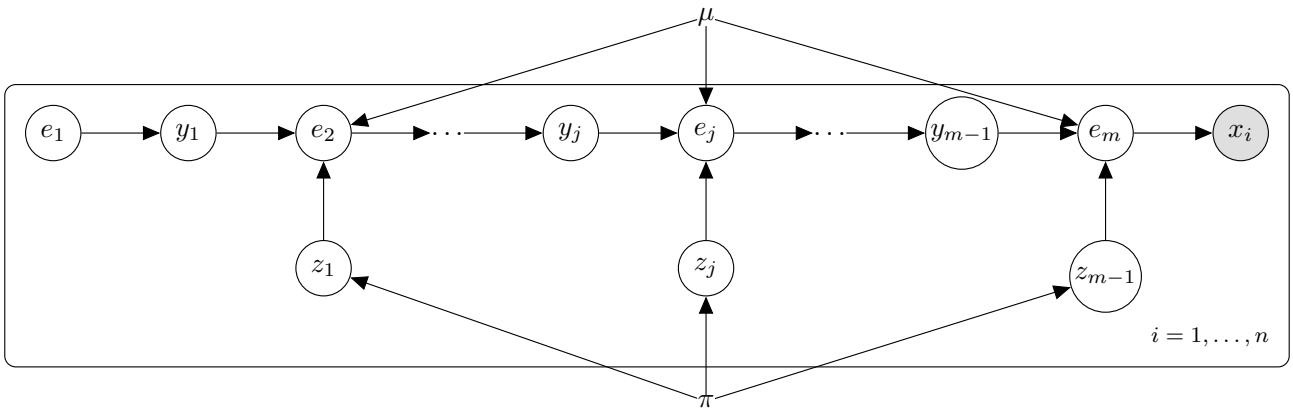

Figure 1: Graphical model of the stochastic Binary Ordinal Search.

2.2 Parameter estimation

EM algorithm. In the univariate case, the underlying BOS distribution of a data sample $(x_1, \ldots, x_n)$ can be estimated using the Expectation-Maximization (EM) algorithm. The main idea behind EM is to iteratively compute the parameters (μ, π) of the distribution to maximize the log-likelihood until convergence when the model is explained by latent variables that are not observed with the data as it is the case with the BOS distribution [**?**].

1. Initialization: The parameters of the distribution are initialized using a pre-defined heuristic. In our case, we initialize $(\mu^{(0)}, \pi^{(0)})$ randomly in their respective range.

2. Expectation step: The latent variables $(c_i)_{i\in\{1,\ldots,n\}}$ are estimated using the current approximation of the parameters $(\mu^{(t)}, \pi^{(t)})$. This consists in computing the posterior probability of every latent variable using the conditional probability formula for all $i \in \{1, \ldots, ..., n\}$

$$\mathbb{P}(c_i|x_i, \mu^{(t)}, \pi^{(t)}) = \frac{\mathbb{P}(c_i, x_i|\mu^{(t)}, \pi^{(t)})}{\mathbb{P}(x_i|\mu^{(t)}, \pi^{(t)})}. \tag{2}$$

This can be computed recursively for the different latent variables using their prior distribution (BOS model):

$$\mathbb{P}(e_{j+1}|\mu^{(t)}, \pi^{(t)}) = \sum_{e\in\mathcal{P}(\{1,\ldots,m\})} \mathbb{P}(e_{j+1}|e_j = e, \mu^{(t)}, \pi^{(t)})\mathbb{P}(e_j = e|\mu^{(t)}, \pi^{(t)}). \tag{3}$$

The intermediate conditional probability is obtained by conditioning on the different values that can be taken by y_j and z_j when $e_j = e$:

$$\mathbb{P}(e_{j+1}|e_j = e|\mu^{(t)}, \pi^{(t)}) = \sum_{y_j\in e} \mathbb{P}(e_{j+1}|e_j = e, y_j, \mu^{(t)}, \pi^{(t)})\mathbb{P}(y_j|e_j = e) \tag{4}$$

$$\begin{aligned} &= \sum_{y_j\in e} (\pi^{(t)}\mathbb{P}(e_{j+1}|e_j, y_j, z_j = 1, \mu^{(t)}, \pi^{(t)}) \\ &\quad + (1 - \pi^{(t)})\mathbb{P}(e_{j+1}|e_j, y_j, z_j = 0, \mu^{(t)}, \pi^{(t)}))\mathbb{P}(y_j|e_j = e). \end{aligned} \tag{5}$$

The components of the sum can be more easily computed by distinguishing between the cases where $z_j = 0$ and $z_j = 1$ with closed-forms expressions from the prior.
Moreover, for y_j, the posterior distribution can also be obtained using the following expression where we also condition on e_j:

$$\mathbb{P}(y_j|\mu^{(t)}, \pi^{(t)}) = \sum_{e\in\mathcal{P}(\{1,\ldots,m\}} \mathbb{P}(y_j|e_j = e, \mu^{(t)}, \pi^{(t)})\mathbb{P}(e_j = e|\mu^{(t)}, \pi^{(t)}). \tag{6}$$

For z_j, $\mathbb{P}(z_j|\mu^{(t)}, \pi^{(t)})$ is a Bernoulli variable of probability $\pi^{(t)}$ so it can easily be expressed:

$$\mathbb{P}(z_j|\mu^{(t)}, \pi^{(t)}) = \begin{cases} \pi^{(t)} & \text{if } z_j = 1 \\ 1 - \pi^{(t)} & \text{if } z_j = 0 \end{cases}. \tag{7}$$

Note that since in the BOS model e_m is identified with x_i, we get the following joint distribution for all the latent variables c_i with the observation x_i:

$$\mathbb{P}(c_i, x_i|\mu^{(t)}, \pi^{(t)}) = \mathbb{P}(e_m|c_i, \mu^{(t)}, \pi^{(t)})\mathbb{P}(c_i|\mu^{(t)}, \pi^{(t)}). \tag{8}$$

3. Maximization step: During this step, the next iteration of the parameters $(\mu^{(t+1)}, \pi^{(t+1)})$ that maximizes the log-likelihood of observing the data taking into account the latent-variables probabilities computed during the Expectation step. As proposed by **?**, only π is updated and μ is fixed during the entire algorithm. This is done for every possible value of μ and the value that maximizes the log-likelihood is chosen.
The new value of π is given after computing the maximizer of the expectation of the log-likelihood in π:

$$\pi^{(t+1)} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{m-1}\mathbb{P}(z_{ij}=1|x_i,\mu^{(t)},\pi^{(t)})}{n(m-1)}. \tag{9}$$

[1]

AECM algorithm. Similarly to the EM algorithm, Alternating Expectation-Conditional Maximization (AECM) [**?**] is separated in two steps. However, in this case, we consider multivariate ordinal data with different possible distributions (clusters) priors for the data. This is done, just like for the Gaussian Mixture Model case, using 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 cluster.

1. Expectation step: In this case, the expectation step consists in just computing the probability for every data point to belong to each cluster:

$$\mathbb{P}(w_{ik}=1|x_i|\alpha^{(t)},\mu^{(t)},\pi^{(t)}) = \frac{\alpha_k^{(t)}\mathbb{P}(x_i|w_{ik}=1,\mu_k^{(t)},\pi_k^{(t)})}{\sum_{l=1}^{p}\alpha_l^{(t)}\mathbb{P}(x_i|w_{il}=1,\mu_l^{(t)},\pi_l^{(t)})}. \tag{10}$$

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

$$\alpha_k^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n}\mathbb{P}(w_{ik}=1|x_i,\alpha^{(t)},\mu^{(t)},\pi^{(t)}). \tag{11}$$

And then the parameters $(\mu_k^{(t+1)}, \pi_k^{(t+1)})$ are updated after using an EM algorithm in the univariate case for every cluster k and for every dimension of the multivariate variables independently using the data on the corresponding dimension.

2.3 Alternative model "Globally Ordered Data" (GOD)

In the article under study, the authors motivated the use of binary search with the following sentence: "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 display 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.

2.3.1 Probabilistic 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 $Z = (Z_1,...,Z_m) \in 0,1^m$ and $C \in 0,1^m$. $(Z_j)_{j=1...m}$ is a vector of independent Bernoulli variables, with parameter π. For $j = 1...m$, 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 results of the comparisons depending on both Z and the parameter μ. It is defined as follows:

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

[1] The authors of the paper did not mention how they computed this formula. We computed it by marginalizing on every possible run of the binary search algorithm, leading to high computational cost.

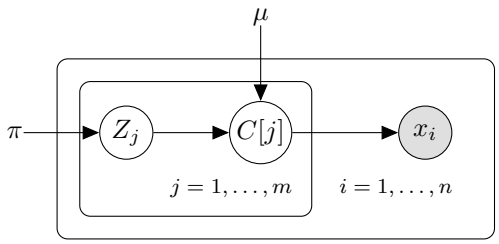

Figure 2: Graphical model of the GOD model.

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 1. The graphical model associated with this probabilistic model is depicted on Figure 2.

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

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

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

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

The proof can be found in appendix 4.

2.3.2 Parameter estimation

We want to estimate π and μ from a sample $X = (x^1, \ldots, x^n) \in [\![1, m]\!]^n$ of n observations of x generated by the GOD model. We aim at maximizing the likelihood of the sample : $\Pr(X|\pi, \mu)$.

We define for $x \in [\![1, m]\!]$,

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

Lemma 1.

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

Proof in appendix 4.

Definition 2. *We define for $x \in [\![1, m]\!], \mu \in [\![1, m]\!], d \in [\![0, m]\!]$:*

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

For a given m, u can be fully computed in $\mathcal{O}(m^2 2^m)$ time. We believe that it might be possible to compute it in polynomial time, but we did not have time to find how. Although still costly, it only needs to be computed once for a given m and can be stored in $\mathcal{O}(m^3)$ space.

Theorem 2 (Data likelihood)**.**

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

$$L_X(\pi, \mu) = \sum_{i=1}^{n} \log \left[\sum_{d=0}^{m} \pi^{m-d} (1 - \pi)^d u(x^i, \mu, d) \right] \tag{12}$$

$$= mn \log \pi + \sum_{i=1}^{n} \log \left[\sum_{d=0}^{m} \left(\frac{1 - \pi}{\pi} \right)^d u(x^i, \mu, d) \right]. \tag{13}$$

Proof. We just use the fact that the random variables $(x^i)_{i=1}^n$ are independent and Theorem 5 from Appendix F. □

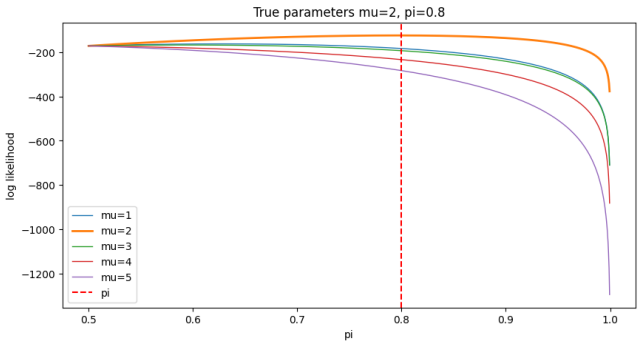

Figure 3: Log-likelihoods for each possible μ for $n = 100$ samples with the true parameters being $m = 5$, $\mu = 2$ and $\pi = 0.8$

Similarly to the BOS model, we estimate $\hat{\pi}$ for each fixed μ. Then we keep the couple $(\hat{\pi}, \mu)$ that maximizes the log-likelihood. To estimate π for a fixed μ, we considered different possibilities. The first option is to use a simple grid search. This is feasible as evaluating L_X is not too costly and can be done in $\mathcal{O}(nm)$ time. Then, we considered using the EM algorithm, similar to the approach taken for the BOS model. We derived the update rules and implemented it. Unfortunately, there was an issue with our EM implementation for this model that we did not succeed in fixing.

We can examine the log-likelihood functions in Figure 3 to understand how to optimize them. As observed, each of the functions $\pi \mapsto L_X(\pi, \mu)$ is concave on the interval $[0.5, 1]$.

Theorem 3. $\forall \mu \in [\![1, m]\!]$,

$$\pi \mapsto L_X(\pi, \mu)$$

is concave on $[0.5, 1]$.

For the proof, see appendix 6.

As the log-likelihood is concave, we can employ the trichotomy method to find the maximum with a few evaluations of L_X. For a precision ε, the trichotomy method requires $k \geqslant \ln \varepsilon \frac{\ln 2}{\ln 2 - \ln 3} = \mathcal{O}(\ln \varepsilon)$ steps and therefore $\mathcal{O}(\ln \varepsilon)$ evaluation of L_X.

Complexity The pre-computational cost of computing u is $\mathcal{O}(m^2 2^m)$.

The evaluation of L_X is done in $\mathcal{O}(m^2)$. We run the trichotomy algorithm m times, and therefore, we evaluate L_X $\mathcal{O}(m \ln \varepsilon)$ times. The total cost is thus $\mathcal{O}(m^2 \ln \varepsilon)$ for m being the number of categories and ε being the precision on π.

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. We finally 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.

3.1 Experimental setup

Parameters of the experiment In this section, we propose to test the BOS model on synthetic data and on 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 designed for this kind of data such as K-Means [**?**] and Gaussian Mixture Models [**?**].

We also test the AECM algorithm for the BOS model with both a random initialization of the parameters and an initialization of the parameters using the K-Means algorithm.

Runtimes are also measured on the same machine for all the algorithms to compare their efficiency.

Dataset. 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...) [**?**]. 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, ...) [**?**]. 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, ...) [**?**]. 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, ...) [**?**]. 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, ...) [**?**]. 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.

Evaluation method. For most of the 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 and to compare different methods more easily.

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 and 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 [**?**] 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 4 in 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 [**?**]. 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.

$$W(\hat{y}, y) = \min_{\gamma \in \Gamma(\hat{y}, y)} \sum_{i,j} \gamma_{i,j} \, ||i - j|| \,, \tag{14}$$

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 [**?**].

$$\text{ARI}(\hat{y}, y) = \frac{\sum_{i,j} \binom{n_{i,j}}{2} - \left[\sum_i \binom{\hat{n}_i}{2} \sum_j \binom{n_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{\hat{n}_i}{2} + \sum_j \binom{n_j}{2}\right] - \left[\sum_i \binom{\hat{n}_i}{2} \sum_j \binom{n_j}{2}\right] / \binom{n}{2}}, \tag{15}$$

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 Estimation methods

In this section, we present the results obtained after running the AECM estimation algorithms for different parameters used to generate data from the BOS distribution and from the GOD model in order to compare the different estimators for their respective distributions. We will then also try the different models on real-life datasets in order to check how they perform for clustering on real data.

BOS distribution. We first test the AECM algorithm on the BOS distribution with experiments similar to **?** in order to confirm that our implementation is coherent with the algorithm proposed and yields results that are close. We generate data from the BOS distribution with different parameters 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. We repeat this process multiple times for different parameters and average the results to obtain the results presented in Table 1 in Appendix A.

GOD model. We then test the AECM algorithm on the GOD model with experiments similar to **?** in order to confirm that the estimation algorithms and the estimators proposed are able to estimate the parameters generated from a GOD distribution. We use data generated from the GOD model with similar parameters as for the BOS distribution. The results are presented in Table 2 in the appendix section.

3.3 Experiments with real-life datasets

In this section, we present the results obtained after running the AECM algorithm on the real-life datasets presented above. We then compare the results obtained with the BOS model with the results obtained with the GOD model and with the results obtained with the K-Means algorithm and the Gaussian Mixture Models. We also compare the results obtained with the two different methods to match the predicted clusters with the true clusters. The results are presented in Table 3 in the appendix B.

We notice that although K-Means allows to significantly reduce the runtime of both the BOS and the GOD models estimations, it does not necessarily increase the clustering score and the classification score. The BOS model, because of its complexity, is also the longest to run but seems to be competitive with the other models on most datasets.

In order to get a better idea of the differences between the clustering methods, we also plot t-SNE visualizations [**?**] for different datasets and the multiple models. Figures 5 in appendix C shows the plots of all the features and the associated true labels and clusters.

Histograms and assignment matrices of some datasets are provided in appendix E and appendix D in order to get a better understanding of the different assignments obtained in these settings.

4 Conclusion

In this study, we analyzed model-based clustering for ordinal data, with a specific focus on the Binary Ordinal Search (BOS) and a proposed alternative we called Globally Ordinal Distribution (GOD) models. We aimed to understand and evaluate their efficiency in clustering and classifying ordinal data compared to more traditional methods like K-Means and Gaussian Mixture Models. Our exploration spanned both synthetic and real-world datasets, providing a comprehensive view of the models' performance in various scenarios.

The experiments on synthetic data confirmed the theoretical foundations of the BOS and GOD models. When parameters were known, both models showed an ability to recover the underlying structure of the generated data.

Particularly, the BOS model, despite its computational intensity due to its design, performed well in clustering tasks, highlighting its potential for applications with ordinal data. The GOD model, with its more manageable computational requirements, also demonstrated promising results, making it a practical alternative for larger datasets.

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.

5 Contribution statement

This project reflects a collaborative effort where all three students, Ali Ramlaoui, Thomas Michel and Théo Rudkiewicz, made equal and substantial contributions. We give here the main but not exclusive focus of each one:

- Ali Ramlaoui focused on the experiments and the implementation of the AECM algorithm.
- Thomas Michel implemented the BOS model and the EM algorithm for the BOS model.
- Théo Rudkiewicz developed the GOD model and the algorithm associated with it.

A AECM tests for BOS and GOD distributions

Init.	n	$n_{clusters}$	d	n_{cats}	$\Delta\alpha$	$\Delta\mu$	$\Delta\pi$
K-Means	50	3	3	2	0.143	0.167	0.295
				3	0.135	0.667	0.116
			5	2	0.090	0.200	0.172
				3	0.083	0.333	0.105
		5	3	2	0.099	0.667	0.337
				3	0.142	0.778	0.356
			5	2	0.099	0.400	0.210
				3	0.125	0.400	0.157
	250	3	3	2	0.111	0.167	0.344
				3	0.021	0.444	0.206
			5	2	0.152	0.200	0.162
				3	0.043	0.267	0.106
		5	3	2	0.149	0.667	0.423
				3	0.315	0.444	0.246
			5	2	0.134	0.400	0.290
				3	0.174	0.467	0.129
Random	50	3	3	2	0.112	0.167	0.074
				3	0.085	0.222	0.025
			5	2	0.011	0.000	0.043
				3	0.039	0.067	0.038
		5	3	2	0.058	0.167	0.081
				3	0.073	0.111	0.079
			5	2	0.095	0.300	0.130
				3	0.068	0.067	0.117
	250	3	3	2	0.095	0.000	0.035
				3	0.018	0.222	0.007
			5	2	0.031	0.000	0.016
				3	0.017	0.067	0.019
		5	3	2	0.037	0.167	0.022
				3	0.057	0.222	0.037
			5	2	0.044	0.000	0.022
				3	0.020	0.067	0.052

Table 1: Results of the experiments for the AECM algorithm no synthetic 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.

Init.	n	$n_{clusters}$	d	n_{cats}	$\Delta\alpha$	$\Delta\mu$	$\Delta\pi$
K-Means	50	3	3	2	0.103	0.167	0.166
				3	0.132	0.444	0.095
			5	2	0.063	0.300	0.088
				3	0.074	0.067	0.020
		5	3	2	0.082	0.333	0.207
				3	0.149	0.778	0.118
			5	2	0.102	0.300	0.122
				3	0.194	0.400	0.089
	250	3	3	2	0.093	0.167	0.152
				3	0.079	0.222	0.076
			5	2	0.088	0.200	0.071
				3	0.121	0.267	0.073
		5	3	2	0.083	0.833	0.211
				3	0.104	0.889	0.125
			5	2	0.097	0.300	0.110
				3	0.113	0.400	0.064
Random	50	3	3	2	0.063	0.167	0.104
				3	0.113	0.333	0.084
			5	2	0.050	0.100	0.057
				3	0.075	0.267	0.053
		5	3	2	0.058	0.333	0.145
				3	0.177	0.667	0.092
			5	2	0.083	0.400	0.112
				3	0.153	0.467	0.056
	250	3	3	2	0.032	0.167	0.093
				3	0.197	0.222	0.042
			5	2	0.014	0.200	0.050
				3	0.072	0.133	0.017
		5	3	2	0.052	0.167	0.118
				3	0.098	0.889	0.146
			5	2	0.058	0.400	0.085
				3	0.164	0.400	0.046

Table 2: Results of the experiments for the AECM algorithm no synthetic 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.

Dataset	Method	Runtime (s)	F1	Accuracy	Wasserstein	ARI
Zoo	**BOS Random**	47.28	0.77	0.78	0.35	**0.76**
	BOS K-Means	4.75	**0.86**	**0.84**	0.17	0.90
	GOD Random	52.64	0.87	0.86	0.23	0.83
	GOD K-Means	15.12	0.87	0.85	**0.14**	0.90
	K-Means	**0.01**	0.70	0.64	0.84	0.58
	Gaussian	0.75	0.79	0.76	0.33	0.73
Car Evaluation	**BOS Random**	481.93	0.46	0.40	0.66	0.02
	BOS K-Means	415.80	0.42	0.37	0.73	-0.01
	GOD Random	28.95	0.43	0.40	0.41	-0.04
	GOD K-Means	19.26	0.46	0.39	0.94	0.05
	K-Means	**0.01**	0.41	0.34	0.91	0.01
	Gaussian	0.04	0.44	0.36	1.07	0.02
Hayes-Roth	**BOS Random**	307.60	0.37	0.39	0.25	0.00
	BOS K-Means	101.11	0.36	0.36	0.30	-0.01
	GOD Random	11.22	0.37	0.39	**0.25**	-0.01
	GOD K-Means	8.49	0.37	0.36	0.23	-0.01
	K-Means	**0.00**	0.34	0.33	0.16	-0.01
	Gaussian	0.02	**0.45**	**0.45**	0.11	**0.07**
Caesarian	**BOS Random**	52.78	0.41	0.56	0.41	-0.01
	BOS K-Means	35.48	0.53	0.53	0.05	-0.01
	GOD Random	3.47	0.54	0.54	**0.04**	-0.01
	GOD K-Means	3.51	0.59	0.59	0.09	0.02
	K-Means	**0.01**	0.56	0.56	0.09	0.00
	Gaussian	0.02	**0.60**	**0.60**	0.00	**0.03**

Table 3: 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 and underlined.

B Metrics on real-world datasets

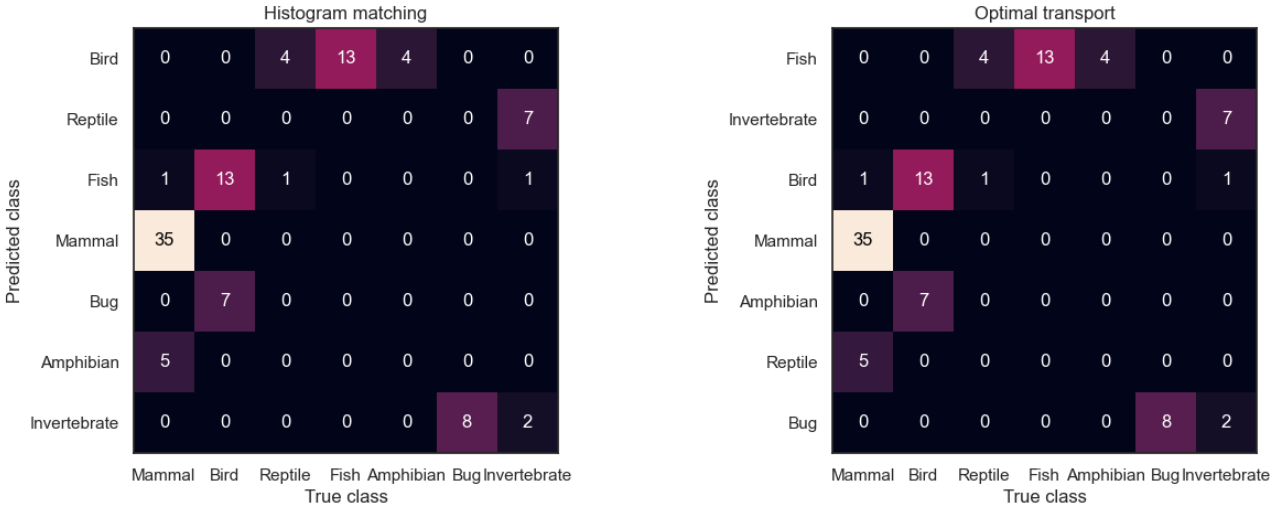

Figure 4: Illustration of the two assignment matrices from the different methods after clustering the Zoo dataset. On the left, the naive method and on the right, the optimal assignment method. The numbers in the matrices represent the number of samples in the predicted cluster i that are in the true cluster j.

C t-SNE plots

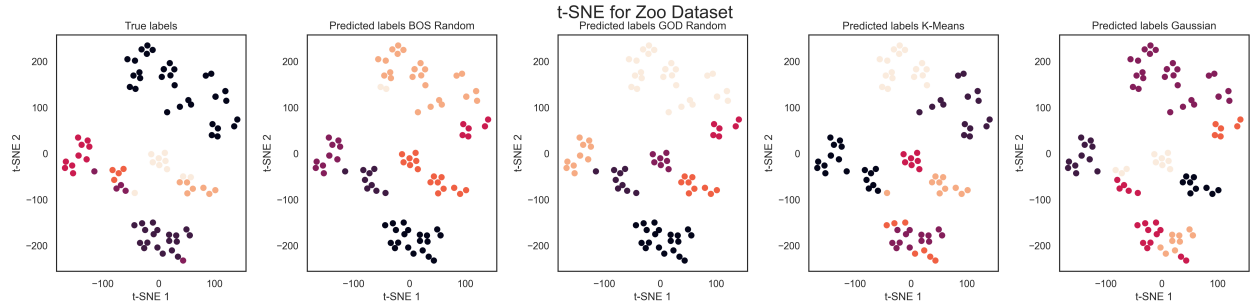

Figure 5: t-SNE visualization of the Zoo dataset with the true labels and the predicted clusters for the BOS model with random initialization, the GOD model with random initialization, the K-Means algorithm and the Gaussian Mixture Models.

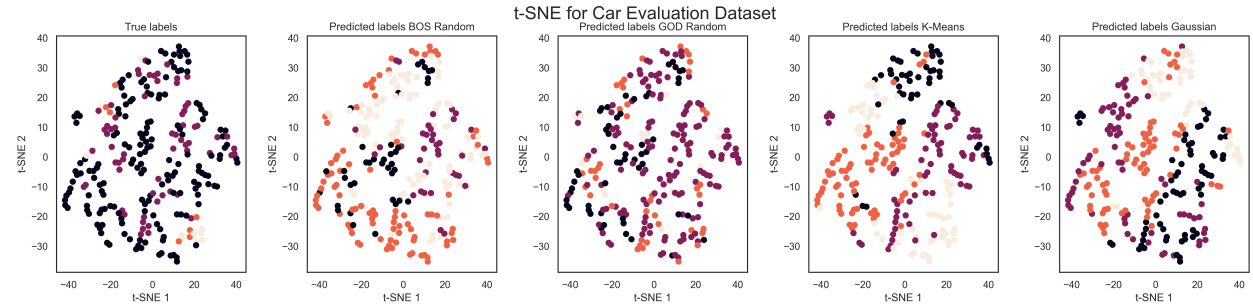

Figure 6: t-SNE visualization of the Car Evaluation dataset with the true labels and the predicted clusters for the BOS model with random initialization, the GOD model with random initialization, the K-Means algorithm and the Gaussian Mixture Models.

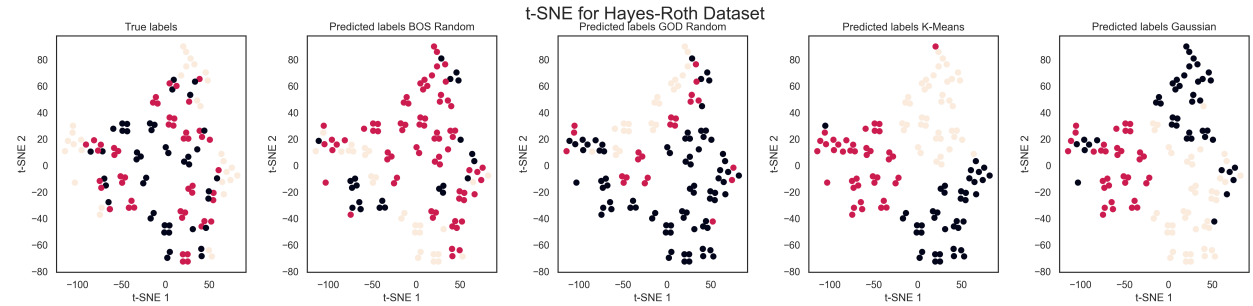

Figure 7: t-SNE visualization of the Hayes-Roth dataset with the true labels and the predicted clusters for the BOS model with random initialization, the GOD model with random initialization, the K-Means algorithm and the Gaussian Mixture Models.

D Assignment Matrices

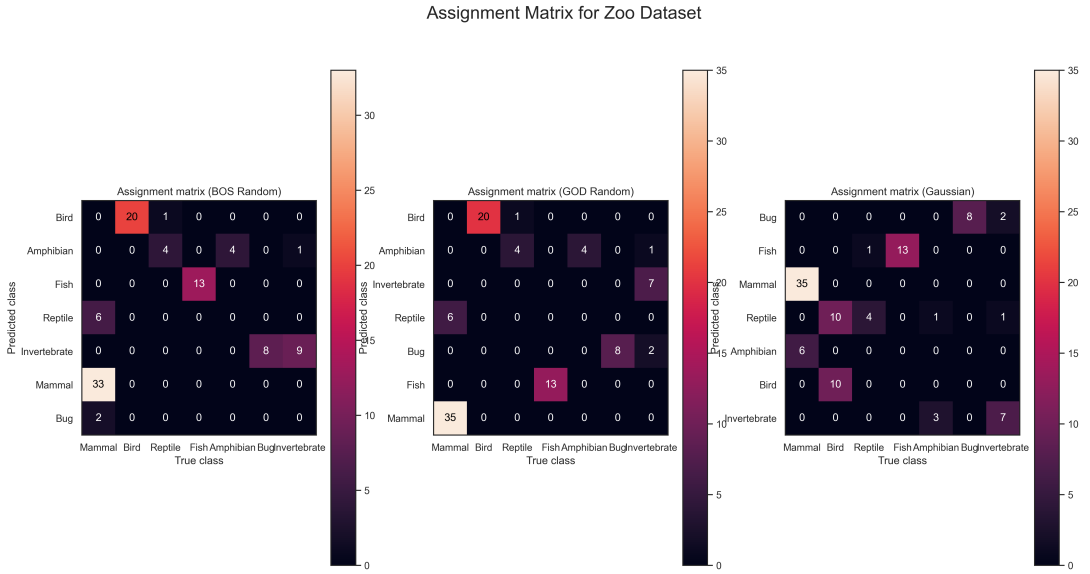

Figure 8: Assignment matrix for the Zoo dataset with different methods.

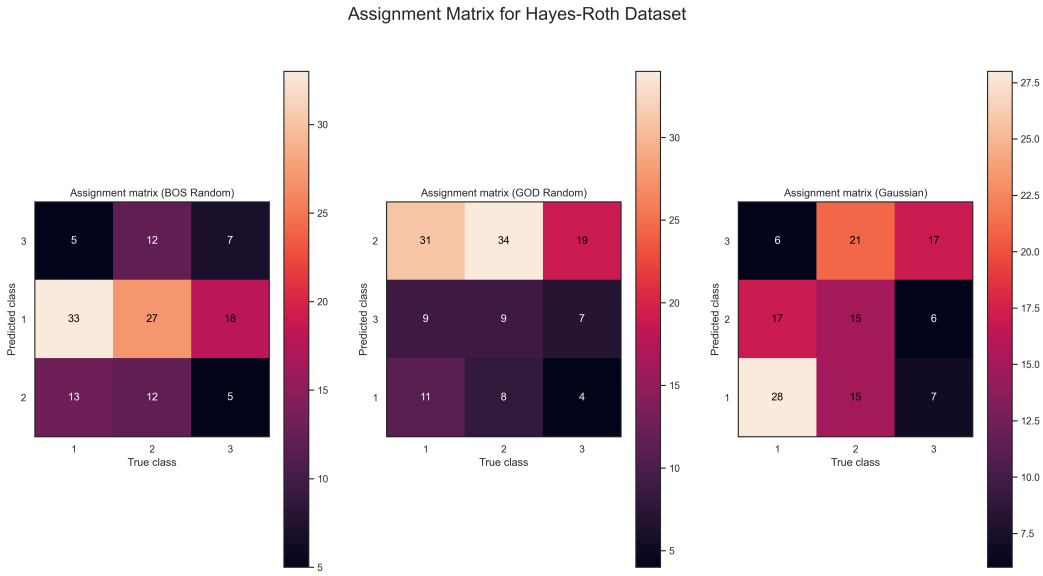

Figure 9: Assignment matrix for the Hayes-Roth dataset with different methods.

E Histograms of the different clustering

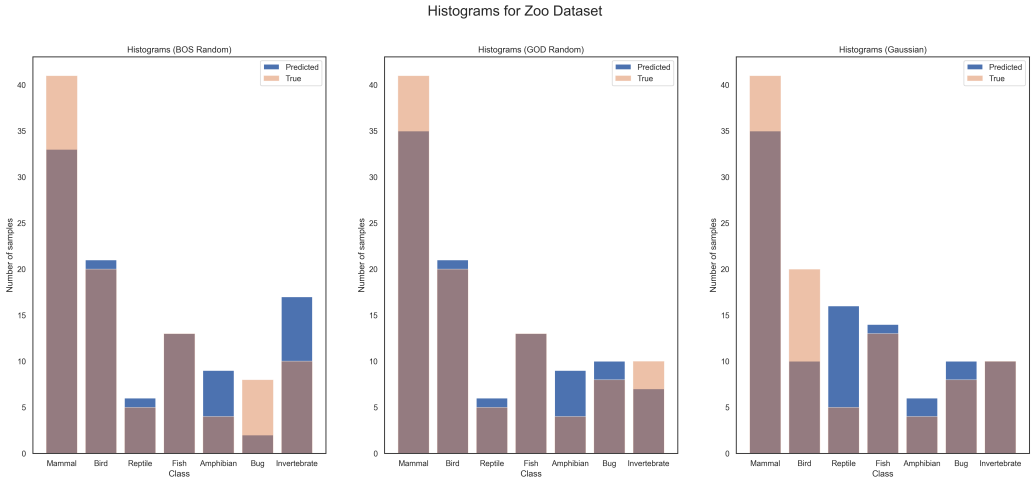

Figure 10: Histograms for the Zoo dataset with different methods.

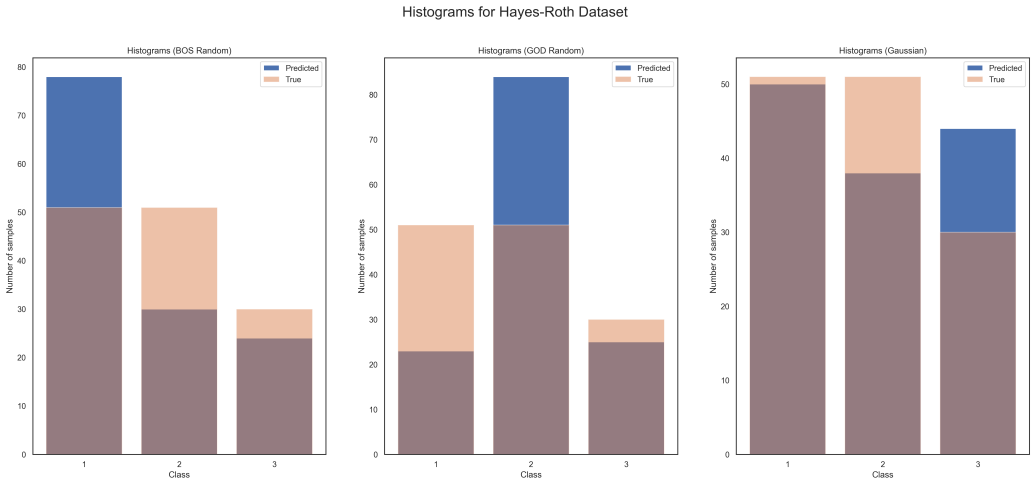

Figure 11: Histograms for the Hayes-Roth dataset with different methods.

F GOD Model proofs

Theorem 4. *If we suppose that the prior distribution of μ is uniform over $[\![1, m]\!]$ and $\pi > \frac{1}{2}$, then $\forall c \in \{0,1\}^m$,*

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

Proof.

Lemma 2.

$$\Pr(C[i] = c[i]|\mu \leqslant i) = c[i]\pi + (1 - c[i])(1 - \pi)$$
$$\Pr(C[i] = c[i]|\mu \nleqslant i) = (1 - c[i])\pi + c[i](1 - \pi)$$

Proof.

$$\Pr(C[i] = c[i] | \mu \leqslant i) \tag{16}$$
$$= \Pr(C[i] = c[i] | Z[i] = 1, \mu \leqslant i) \Pr(Z[i] = 1) + \Pr(C[i] = c[i] | Z[i] = 0, \mu \leqslant i) \Pr(Z[i] = 0) \tag{17}$$
$$= c[i] \Pr(Z[i] = 1) + (1 - c[i]) \Pr(Z[i] = 0) \tag{18}$$
$$= c[i]\pi + (1 - c[i])(1 - \pi) \tag{19}$$
$$\Pr(C[i] = c[i] | \mu \not\leqslant i) \tag{20}$$
$$= \Pr(C[i] = c[i] | Z[i] = 1, \mu \not\leqslant i) \Pr(Z[i] = 1) + \Pr(C[i] = c[i] | Z[i] = 0, \mu \not\leqslant i) \Pr(Z[i] = 0) \tag{21}$$
$$= (1 - c[i]) \Pr(Z[i] = 1) + c[i] \Pr(Z[i] = 0) \tag{22}$$
$$= (1 - c[i])\pi + c[i](1 - \pi) \tag{23}$$

□

Lemma 3. $\forall c \in \{0,1\}^m, \forall k \in [\![1, m]\!]$,

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

Proof. Let us compute for $i \in [\![1, m]\!]$, $\Pr(C = c | \mu = i)$ as the $C[i]|\mu$ are independent and using the previous lemma:

$$\Pr(C = c | \mu = k) = \prod_{i=1}^{m} \Pr(C[i] = c[i] | \mu = k) \tag{24}$$
$$= \prod_{i=1}^{k} \Pr(C[i] = c[i] | \mu \leqslant i) \prod_{i=k+1}^{m} \Pr(C[i] = c[i] | \mu \not\leqslant i) \tag{25}$$
$$= \prod_{i=1}^{k} [c[i]\pi + (1 - c[i])(1 - \pi)] \prod_{i=k+1}^{m} [(1 - c[i])\pi + c[i](1 - \pi)] \tag{26}$$
$$= \pi^{\sum_{i=1}^{k} c[i]} (1 - \pi)^{\sum_{i=1}^{k} (1 - c[i])} \pi^{\sum_{i=k+1}^{m} (1 - c[i])} (1 - \pi)^{\sum_{i=k+1}^{m} c[i]} \tag{27}$$
$$= \pi^{\sum_{i=1}^{k} c[i] + \sum_{i=k+1}^{m} (1 - c[i])} (1 - \pi)^{\sum_{i=1}^{k} (1 - c[i]) + \sum_{i=k+1}^{m} c[i]} \tag{28}$$
$$= \pi^{m - \left[\sum_{i=1}^{k} (1 - c[i]) + \sum_{i=k+1}^{m} c[i]\right]} (1 - \pi)^{\sum_{i=1}^{k} (1 - c[i]) + \sum_{i=k+1}^{m} c[i]} \tag{29}$$
$$= \pi^{m - ||E_k - c||_1} (1 - \pi)^{||E_k - c||_1} \tag{30}$$

□

$$\Pr(\mu = k | C = c) = \frac{\Pr(C = c | \mu = k) \Pr(\mu = k)}{\Pr(C = c)} \tag{31}$$
$$= \frac{\Pr(C = c | \mu = k) \Pr(\mu = k)}{\sum_{i=1}^{m} \Pr(C = c | \mu = i) \Pr(\mu = i)} \tag{32}$$

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

$$\Pr(\mu = k | C = c) = \frac{\Pr(C = c | \mu = k)}{\sum_{i=1}^{m} \Pr(C | \mu = i)} \tag{33}$$

using Lemma 3:

$$\Pr(\mu = k | C = c) = \frac{\pi^{m - ||c - E_k||_1} (1 - \pi)^{||c - E_k||_1}}{\sum_{i=1}^{m} \pi^{m - ||c - E_i||_1} (1 - \pi)^{||c - E_i||_1}} \tag{34}$$
$$\tag{35}$$

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

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

□

Lemma 4.

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

Proof. Using Bayes' theorem, then Lemma 3 and the fact that μ is uniformly distributed over the set defined by the argmin, we have:

$$\begin{aligned}
\Pr(x, C=c|\pi,\mu) &= \Pr(x|c,\pi,\mu)\Pr(C=c|\pi,\mu) && (36)\\
&= \mathbb{1}_{\mathcal{C}_x}(c)\Pr(x|c\in\mathcal{C}_x,\pi,\mu)\Pr(c|\pi,\mu) && (37)\\
&= \mathbb{1}_{\mathcal{C}_x}(c)\frac{\pi^{m-||c-E_\mu||_1}(1-\pi)^{||c-E_\mu||_1}}{\left|\operatorname{argmin}_{k\in[\![1,m]\!]}||c-E_k||_1\right|} && (38)\\
&= \pi^m\mathbb{1}_{\mathcal{C}_x}(c)\frac{\left(\frac{1-\pi}{\pi}\right)^{||c-E_\mu||_1}}{\left|\operatorname{argmin}_{k\in[\![1,m]\!]}||c-E_k||_1\right|} && (39)
\end{aligned}$$

□

Theorem 5 (Observation likelihood)**.**

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

Proof. Using the previous lemma, we have:

$$\begin{aligned}
\Pr(x|\pi,\mu) &= \sum_{c\in\{0,1\}^m}\Pr(x,c|\pi,\mu) && (40)\\
&= \pi^m\sum_{c\in\mathcal{C}_x}\left(\frac{1-\pi}{\pi}\right)^{||c-E_\mu||_1}\left|\operatorname*{argmin}_{k\in[\![1,m]\!]}||c-E_k||_1\right|^{-1} && (41)\\
&= \pi^m\sum_{d=0}^{m}\left(\frac{1-\pi}{\pi}\right)^d\sum_{c\in\mathcal{C}_x/||c-E_\mu||_1=d}\left|\operatorname*{argmin}_{k\in[\![1,m]\!]}||c-E_k||_1\right|^{-1} && (42)
\end{aligned}$$

□

Theorem 6. $\forall\mu\in[\![1,m]\!]$,

$$\pi\mapsto L_X(\pi,\mu)$$

is concave on $[0.5, 1]$.

Sketch of proof. We use the following expression:

$$L_X(\pi,\mu) = mn\log\pi + \sum_{i=1}^{n}\log\left[\sum_{d=0}^{m}\left(\frac{1-\pi}{\pi}\right)^d u(x^i,\mu,d)\right]$$

We use the following result:
For $f(x)=\ln g(x)$ with g twice differentiable, f is concave if and only if $g'(x)^2 - g(x)g''(x)\geqslant 0$ for all x.
We can verify this condition on $x\mapsto\left(\frac{1-x}{x}\right)^d$ for $d\in\mathbb{N}$. Then using the fact that $u\geqslant 0$, we can conclude that:

$$\pi\mapsto\log\left[\sum_{d=0}^{m}\left(\frac{1-\pi}{\pi}\right)^d u(x^i,\mu,d)\right]$$

is concave. Then we can conclude that $\pi\mapsto L_X(\pi,\mu)$ is concave. □