APPLICATION OF THE INFORMATION-THEORETIC CLUSTERING TO THE DISCRETE CHOICE

Alexandre Monti

Supervised by Pavel Ilinov, Evangelos Paschalidis and Professor Michel Bierlaire

Contents

1 Introduction

1.1 Basics of information theory

Before diving into different clustering algorithms using methods of information theory, we need to define the important concepts that we will use later on this paper.

Definition: Let X be a discrete random variable taking values in A_X with distribution p. The entropy of X is given by

$$H(X) = \sum_{x \in A_X} p(x) \cdot \log\left(\frac{1}{p(x)}\right)$$

with the convention that if $p(x) = 0$, then $0 \cdot \log\left(\frac{1}{0}\right) = 0$. The unit for the entropy is the bit.

The entropy captures the average information content (or amount of surprise) associated with the outcomes of a random variable. The entropy is maximum when the distribution is uniform and is zero if an outcome happens with probability 1.

Definition: Let X and Y be two discrete random variables taking values in A_X and A_Y respectively and with joint distribution p. The joint entropy of X and Y is given by

$$H(X,Y) = \sum_{x \in A_X, y \in A_Y} p(x,y) \cdot \log\left(\frac{1}{p(x,y)}\right)$$

One property of the joint entropy is that $H(X) + H(Y) \geq H(X,Y)$ with equality if and only if $X \perp\!\!\!\perp Y$. The intuition behind the joint entropy is closely similar to the one for the entropy. It gives the average information content associated with the outcomes of a random vector.

Definition: The conditional entropy of X given Y is

$$H(X \mid Y) = \sum_{y \in A_Y} p(y) \left[\sum_{x \in A_X} p(x \mid y) \log\left(\frac{1}{p(x \mid y)}\right)\right] = \sum_{x \in A_X, y \in A_Y} p(x,y) \log\left(\frac{1}{p(x \mid y)}\right)$$

The conditional entropy quantifies the average uncertainty about x when y is known.

We can link these different definitions together by the following property:

$$H(X,Y) = H(X) + H(Y \mid X) = H(Y) + H(X \mid Y)$$

Furthermore, $H(X \mid Y) \leq H(X)$.

We could ask ourselves: is there a way to measure how much information I can learn about one variable by observing the other? This is exactly the question that mutual information answers.

Definition: The mutual information between X and Y is given by

$$I(X;Y) = H(X) - H(X \mid Y)$$

With the properties about conditional and joint entropy from above, we know that $I(X;Y) \geq 0$ and $I(X;Y) = I(Y;X)$. Indeed,

$$\begin{aligned} I(X;Y) &= H(X) - H(X \mid Y) \\ &= H(X) + H(Y) - H(X,Y) \\ &= H(X) + H(Y) - H(X) - H(Y \mid X) \\ &= H(Y) - H(Y \mid X) = I(Y;X) \end{aligned}$$

Remark: We can generalize all of the above to continuous random variables easily by replacing sums with integrals. In the continuous case, the entropy can be infinitely large and positive or negative.

Another interesting property of the mutual information is that we can directly link it to the Kullback-Leibler divergence. We first recall what the Kullback-Leibler divergence is:

Definition: Let p and q be two discrete probability distributions defined on the same sample space X. The Kullback-Leibler divergence (or relative entropy) from q to p is defined by

$$D_{KL}(p \mid\mid q) = \sum_{x \in X} p(x) \log \left(\frac{p(x)}{q(x)} \right)$$

Now, if we go back to the definition of mutual information, we have that

$$\begin{aligned} I(X;Y) &= H(X) - H(X \mid Y) \\ &= \sum_{x \in A_X} p(x) \cdot \log \left(\frac{1}{p(x)} \right) - \sum_{x \in A_X, y \in A_Y} p(x,y) \log \left(\frac{1}{p(x \mid y)} \right) \\ &= \sum_{x \in A_X, y \in A_Y} p(x,y) \cdot \log \left(\frac{1}{p(x)} \right) - \sum_{x \in A_X, y \in A_Y} p(x,y) \log \left(\frac{1}{p(x \mid y)} \right) \\ &= \sum_{x \in A_X, y \in A_Y} p(x,y) \cdot \left[\log \left(\frac{1}{p(x)} \right) - \log \left(\frac{1}{p(x \mid y)} \right) \right] \\ &= \sum_{x \in A_X, y \in A_Y} p(x,y) \cdot \log \left(\frac{p(x \mid y)}{p(x)} \right) \\ &= \sum_{x \in A_X, y \in A_Y} p(x,y) \cdot \log \left(\frac{p(x,y)}{p(x) \cdot p(y)} \right) \\ &= D_{KL}(p(x,y) \mid\mid p(x) \cdot p(y)) \end{aligned}$$

Thus, we have that $I(X,Y) = D_{KL}(p(x,y) \mid\mid p(x) \cdot p(y))$.

With these few definitions, we are now ready to delve into the information-theoretic clustering methods that will be the main topic of this work.

1.2 Information Bottleneck method (IB)

Developed by Naftali Tishby, Fernando C. Pereira, and William Bialek in 1999 [1], the information bottleneck method is a powerful framework in machine learning and information theory that aims to extract important information from data while eliminating any irrelevant or duplicated elements. More precisely, the goal of this method is to find a concise representation of the input data X while preserving the relevant information that X gives about the output data Y.

To this aim, we need to solve the following optimization problem: let T be the compressed input data. Given the joint distribution $p(x, y)$, the optimized encoding distribution $q(t|x)$ corresponds to

$$\begin{aligned}\min_{q(t|x)} L[q(t|x)] &= I(X;T) - \beta I(T;Y) \\ &= \sum_{x,t} q(x,t) \cdot \log\left(\frac{q(x,t)}{p(x)q(t)}\right) - \beta \sum_{t,y} q(t,y) \cdot \log\left(\frac{q(t,y)}{q(t)p(y)}\right) \\ &= \sum_{x,t} q(t|x)p(x) \cdot \log\left(\frac{q(t|x)}{q(t)}\right) - \beta \sum_{t,y} q(y|t)q(t) \cdot \log\left(\frac{q(y|t)}{p(y)}\right)\end{aligned}$$

with the additional Markov constraint $T \longleftrightarrow X \longleftrightarrow Y$. This constraint ensures that T can only have information about Y through X.

The idea behind this optimization problem is that the first term pushes for compression, while the second term emphasizes the importance of keeping the relevant information. In this context, β serves as the Lagrange multiplier associated with the constraint of retaining meaningful information. We denote by p the fixed distributions and by q the distributions that we can change or choose.

By making variational calculus, we find that this problem has a formal solution [1] given by:

$$\begin{aligned}q(t|x) &= \frac{q(t)}{Z(x,\beta)} \exp[-\beta D_{\mathrm{KL}}[p(y|x) \mid\mid q(y|t)]] \\ q(y|t) &= \frac{1}{q(t)} \sum_x q(t|x)p(x,y) \\ Z(x,\beta) &\equiv \exp\left[-\frac{\lambda(x)}{p(x)} - \beta \sum_y p(y|x) \log \frac{p(y|x)}{p(x)}\right]\end{aligned}$$

where $D_{\mathrm{KL}}(P(x) \mid\mid Q(x)) = \sum_{x \in X} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$ denotes the Kullback-Leibler divergence. The $\lambda(x)$ in the definition of $Z(x, \beta)$ is the Lagrange multiplier for the normalization of the conditional distributions $q(t \mid x)$ at each x.

This solution is only formal because the first two equations depend on each other which makes them impossible to compute. But with an iterative approach, we can compute a solution which converges to the formal solution. This iterative algorithm works as follows:

Choose some initial distribution $q^{(0)}(t \mid x)$. Compute

$$q^{(0)}(t) = \sum_x p(x) q^{(0)}(t \mid x) \quad \text{and} \quad q^{(0)}(y \mid t) = \frac{1}{q^{(0)}(t)} \sum_x p(x,y) q^{(0)}(t \mid x)$$

The n-th iteration of the algorithm is given by:

$$\begin{aligned}
d^{(n-1)}(x,t) &\equiv D_{\mathrm{KL}}\left[p(y \mid x) \mid q^{(n-1)}(y \mid t)\right] \\
q^{(n)}(t \mid x) &= \frac{q^{(n-1)}(t)}{Z(x,\beta)} \exp\left[-\beta d^{(n-1)}(x,t)\right] \\
q^{(n)}(t) &= \sum_x p(x) q^{(n)}(t \mid x) \\
q^{(n)}(y \mid t) &= \frac{1}{q^{(n)}(t)} \sum_x q^{(n)}(t \mid x) p(x,y)
\end{aligned}$$

1.3 Deterministic Information Bottleneck (DIB)

One of the issue of the IB method is that it produces soft clustering which means that a given input can have multiple outputs with given probabilities. The idea behind the Deterministic Information Bottleneck method (DIB) [2] is to produce hard clustering, i.e. one input has a unique output. To this aim, we modify the cost function of the IB method as follows:

$$\min_{q(t|x)} L[q(t|x)] = H(T) - \beta I(T;Y)$$

still with the same Markov constraint $T \longleftrightarrow X \longleftrightarrow Y$.

How can we interpret the difference between IB and DIB methods? The disparity arises from a modification in the first term where $I(X;T)$ shifts to $H(T)$. In the domain of channel coding, $I(X;T)$ represents compression and suggests a limitation on communication cost between X and T. However, by adopting a source coding perspective, we can substitute $I(X;T)$ with $H(T)$, which denotes the representational cost of T. We can also see the difference between the two methods by substracting one to another:

$$\begin{aligned}
L_{IB} - L_{DIB} &= I(X;T) - \beta I(T;Y) - H(T) + \beta I(T;Y) \\
&= I(X;T) - H(T) = I(T;X) - H(T) \\
&= H(T) - H(T \mid X) - H(T) \\
&= -H(T \mid X)
\end{aligned}$$

This simple calculation reveals that the IB method promotes stochasticity in the encoding distribution $q(t \mid x)$. Indeed, $H(T \mid X)$ represents the stochasticity in the mapping from X to T.

In order to find a solution, we cannot use the same calculus method as for the IB problem. Rather, we

define the following family of cost functions:

$$L_\alpha \equiv H(T) - \alpha H(T \mid X) - \beta I(T;Y)$$

It is trivial to see that $L_{IB} = L_1$. We could say the same for $L_{DIB} = L_0$ but we will use a different strategy in order to find a solution for the DIB method. Indeed, we define the DIB solution as

$$q_{DIB}(t \mid x) = \lim_{\alpha \to 0} q_\alpha(t \mid x)$$

We can now use the same variational approach as for the IB method to find a formal solution [2] which is given by:

$$\begin{aligned}
d_\alpha(x,t) &\equiv D_{\mathrm{KL}}[p(y \mid x) \mid q_\alpha(y \mid t)] \\
l_{\alpha,\beta}(x,t) &\equiv \log(q_\alpha(t)) - \beta d_\alpha(x,t) \\
q_\alpha(t \mid x) &= \frac{1}{Z(x,\alpha,\beta)} \exp\left[\frac{1}{\alpha} l_{\alpha,\beta}(x,t)\right] \\
q_\alpha(y \mid t) &= \frac{1}{q_\alpha(t)} \sum_x q_\alpha(t \mid x) p(x,y)
\end{aligned}$$

where $Z(x,\alpha,\beta)$ is a normalization factor given by

$$\begin{aligned}
z(x,\alpha,\beta) &= \frac{1}{\alpha} - 1 + \frac{\lambda(x)}{\alpha p(x)} + \frac{\beta}{\alpha} \sum_y p(y \mid x) \log\left(\frac{p(y \mid x)}{p(y)}\right) \\
Z(x,\alpha,\beta) &= \exp[-z(x,\alpha,\beta)]
\end{aligned}$$

Now that we have a general formal solution, we can take the limit $\alpha \to 0$ to finally have a formal solution for the DIB problem. By computing the limit, we find that

$$\lim_{\alpha \to 0} q_\alpha(t \mid x) = f : X \to T$$

where

$$\begin{aligned}
f(x) \equiv t^* &= \arg\max_t [l(x,t)] \\
l(x,t) &\equiv \log(q(t)) - \beta D_{\mathrm{KL}}[p(y \mid x) \mid q(y \mid t)]
\end{aligned}$$

More explicitly, for a fixed x the limit of $q_\alpha(t \mid x)$ when $\alpha \to 0$ becomes a delta function which is 1 at the value of t which maximizes $l(x,t)$. This is why we call this method the deterministic information bottleneck as the solution is a deterministic encoding distribution. The term $\log(q(t))$ in the equation promotes minimizing the number of unique values for t, favoring assigning each x to a t that already has many other x values associated with it. Meanwhile, the KL divergence term, similar to the original IB problem, ensures that the chosen t values retain as much information from x about y as possible.

The parameter β has the same role as in the original problem and controls the balance between the importance placed on compression and prediction.

As before, the solution for the DIB problem is only formal and we need to use an iterative algorithm in order to be able to find a computable solution. The iterative algorithm works as follows.

Choose some initial distribution for $f^{(0)}(x)$. Set

$$q^{(0)}(t) = \sum_{x:f^{(0)}(x)=t} p(x) \quad \text{and} \quad q^{(0)}(y \mid t) = \frac{\sum_{x:f^{(0)}(x)=t} p(x,y)}{\sum_{x:f^{(0)}(x)=t} p(x)}$$

Then apply the following steps until convergence:

$$\begin{aligned}
d^{(n-1)}(x,t) &\equiv D_{\mathrm{KL}}\left[p(y \mid x) \mid q^{(n-1)}(y \mid t)\right] \\
l_\beta^{(n-1)}(x,t) &\equiv \log(q^{(n-1)}(t)) - \beta d^{(n-1)}(x,t) \\
f^{(n)}(x) &= \arg\max_t [l_\beta^{(n-1)}(x,t)] \\
q^{(n)}(t \mid x) &= \delta\left(t - f^{(n)}(x)\right) \\
q^{(n)}(t) &= \sum_x p(x) q^{(n)}(t \mid x) = \sum_{x:f^{(n)}(x)=t} p(x) \\
q^{(n)}(y \mid t) &= \frac{1}{q^{(n)}(t)} \sum_x q^{(n)}(t \mid x) p(x,y) = \frac{\sum_{x:f^{(n)}(x)=t} p(x,y)}{\sum_{x:f^{(n)}(x)=t} p(x)}
\end{aligned}$$

The intuition behind this algorithm is the following:

- Calculate the Kullback-Leibler (KL) divergence between the true conditional distribution $p(y \mid x)$ and the current estimated conditional distribution $q^{(n-1)}(y \mid t)$. The KL divergence tells us how much $p(y \mid x)$ is "close" to $q^{(n-1)}(y \mid t)$. If $D_{KL} = 0$, the two distributions are the same.
- Calculate $l_\beta^{(n-1)}(x,t) = \log(q^{(n-1)}(t)) - \beta \cdot d^{(n-1)}(x,t)$, which combines compression and relevance terms using the Lagrange multiplier β.
- Assign each data point x to the cluster t that maximizes $l_\beta^{(n-1)}(x,t)$.
- Update the conditional distribution of clusters given data points $q^{(n)}(t \mid x)$ by setting it to a Dirac delta function $\delta(t - f^{(n)}(x))$.
- Update the marginal distribution of clusters $q^{(n)}(t)$ by summing over all data points assigned to each cluster.
- Update the conditional distribution of data points given clusters $q^{(n)}(y \mid t)$ by computing the ratio of the joint distribution $p(x,y)$ to the marginal distribution $p(x)$ for each cluster.

2 Implementation of the IB and DIB methods

2.1 Geometric DIB algorithm

Firstly, we'll focus on implementing the DIB method specifically tailored for geometric clustering. This approach will help us to understand the underlying dynamics and behavior of the algorithm.

We consider two dimensions data points $\{\mathbf{x}_i\}_{i=1:n}$ and we want to cluster them. Here, we need to choose what is X and what is Y. We first make the same choice as in [3] where X is the data point index i and Y is the data point location $\mathbf{x}$. Thus, we want to cluster data indices i in a way that keeps as much information about data location as possible.

We need now to choose the joint distribution $p(i, \mathbf{x}) = p(i)p(\mathbf{x} \mid i)$. It is trivial to choose $p(i) = \frac{1}{n}$ but the choice for $p(\mathbf{x} \mid i)$ is more complicated. Following [3], we take

$$p(\mathbf{x} \mid i) \propto \exp\left[-\frac{1}{2s^2} d(\mathbf{x}, \mathbf{x}_i)\right]$$

where s corresponds to the units of distance and d is a distance metric, for example the Euclidean distance $d(\mathbf{x}, \mathbf{x}_i) = ||\mathbf{x} - \mathbf{x}_i||^2$.

One important observation is that if $q^{(n)}(t) = 0$, which means that the cluster t is empty, then this cluster will remain empty until convergence of the algorithm. Indeed, $\log\left(q^{(n)}(t)\right) = -\infty \implies l_\beta^m(x,t) = -\infty$ for all $m \geq n$. Thus, the number of non-empty clusters $|T|$ can only decrease during the computation of the algorithm. We then make the choice of initializing each point as its own cluster.

After multiple attempts, a persistent issue arises: the algorithm becomes stagnant from the first step, with distributions remaining constant throughout iterations. By investigating and reading again the different papers about information bottleneck [1][2][3], we understand what is going on. Indeed, the objective function

$$L_{DIB} = \min_{q(t|x)} H(T) - \beta I(T;Y)$$

is non-convex, leading to the possibility of converging to a local minimum rather than the global minimum. This phenomenon is particularly evident when initializing each point as its own cluster. At the initialization step, the distribution of x (or i in the case of geometric clustering) is uniform, resulting in $q(y|t)$ being identical to $p(y|x)$ when x is assigned to t, which leads to $d^{(0)}(x,t) = 0$ when x is assigned to t and $d^{(0)}(x,t) > 0$ otherwise. Consequently, the algorithm becomes trapped in a local minimum because $\arg\max_t \left[l_\beta^{(n)}(x,t)\right] = t_x$ for all $n \geq 0$ where t_x represents the cluster to which x is initially assigned. In other words, each point remains in its own cluster throughout the computation.

To overcome the issue of being trapped in a local minimum, we apply the same methodology as in [3]: at each iteration of the algorithm, we assess whether the objective function L_{DIB} could be minimized further by merging two clusters. By doing this, we ensure that the algorithm will reach the global minimum.

These improvements in the original algorithm lead to great results. For the first example, we generate

90 data points from three different mutivariate gaussian distributions with covariance matrix $\frac{1}{10}I_2$ and mean $(3,0)$, $(0,4)$ and $(5,5)$ respectively. We run the algorithm for 100 iterations and we vary the free parameter β. The result of the clustering is shown below:

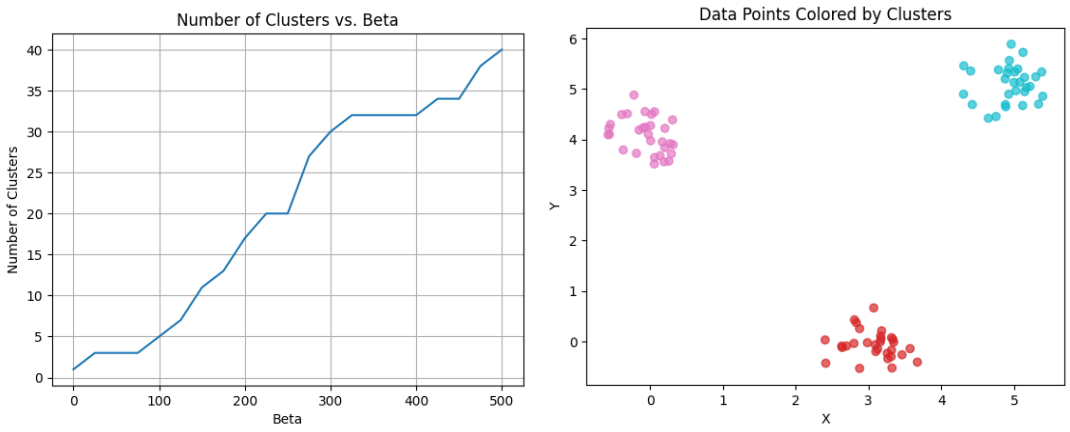

Figure 1: I WILL RECOMPUTE THE FIRST IMAGE WITH MORE BETA VALUES Left: Number of clusters determined by the DIB algorithm against β for 50 iterations. Right: Data points colored by clusters using the DIB algorithm with $|X| = 90$, $\beta = 3.7$ and 100 iterations.

We can see on the first plot that the algorithm is quite robust to determine the three clusters. Indeed, for $\beta \in [2; 90]$ the algorithm finds them with only 50 iterations. On the second plot, we can verify that the three clusters are the true clusters and not some random association between points.

With the algorithm in hand, we can now compute an IB curve or in our case, a DIB curve. A DIB curve illustrates the tradeoff between compression and prediction. It plots the mutual information $I(T;Y)$ between the cluster representation T and the target variable Y against the entropy of the cluster representation T, $H(T)$. This curve provides insights into how much information about the target variable Y can be retained in the cluster representation T, given a certain level of compression. Solutions positioned below the DIB curve are inherently suboptimal.

However, the DIB framework does not prescribe a method for selecting a single solution from the array of solutions along its boundary. Intuitively, when confronted with a Pareto-optimal boundary [1], targeting a solution at the curve's "knee" appears to be advantageous. This "knee" point represents the maximum magnitude second derivative of the curve. In extreme scenarios, where the curve exhibits a kink, the second derivative could be infinite, highlighting significant points of interest.

In the case of DIB, where hard clustering is enforced, kinks inevitably appear. This arises because by constraining $q(t \mid x)$ to be a binary matrix, where elements are either 0 or 1, both $q(t)$ and $q(y \mid t)$ can only take a limited set of values. This limitation stems from the definitions $q(t) = \sum_x q(t \mid x)p(x)$ and $q(y \mid t) = \frac{1}{q(t)} \sum_x q(t \mid x)p(x, y)$, where $p(x)$ and $p(x, y)$ are given. Consequently, the entropy $H(T)$ and the mutual information $I(T;Y)$ will attain only a finite number of values, leading to the emergence

[1] A Pareto-optimal boundary represents the set of solutions where it's impossible to improve one objective without seeing a degradation in another.

of kinks in the DIB curve. These kinks signify solutions applicable across a broad range of β values. Consequently, these kinks denote solutions which are robust to variations in model hyperparameters. Such solutions are likely to capture genuine underlying structures within the dataset.

For example, we can compute the DIB curve for our dataset of multivariate gaussian data points.

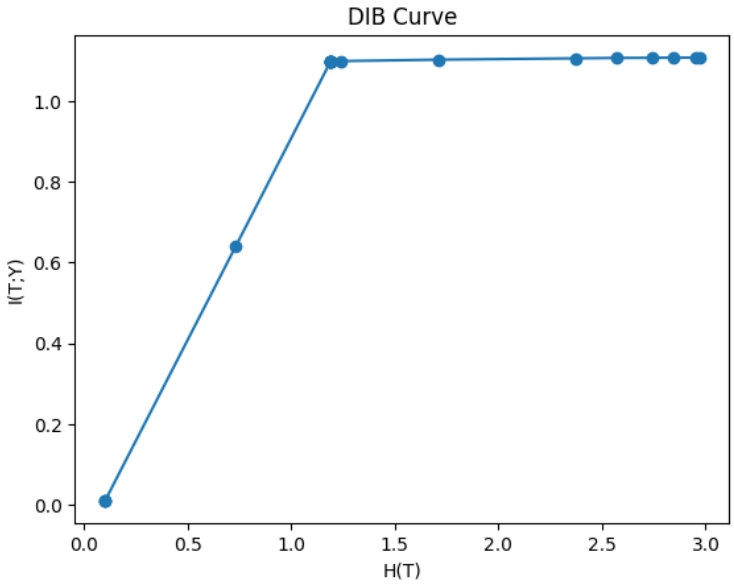

Figure 2: DIB curve for the multivariate gaussian dataset. β takes its values in a uniform vector of 11 points between 0 and 1 and a uniform vector of 9 points between 4 and 500. We fix the number of iterations for each β to 50.

In this plot, the presence of a kink is evident and it highlights the algorithm's robustness to variations in β. Specifically, for β values in at least $[0.4, 60]$, the DIB algorithm identifies three clusters within the dataset. This observation shows us how the DIB curve facilitates the determination of the optimal number of clusters in a given dataset which could be very useful for the following of our work.

[MAYBE WRITE ABOUT THE KINK ANGLE ?]

2.2 General IB algorithm

Now that we have a better overview of what the deterministic information bottleneck works for geometric clustering, we implement the general information bottleneck algorithm based on Algorithm 1 of [2]. The objective is to determine whether we can replicate the IB curve in Figure 1 outlined in the paper. After a few trials, the first results appear:

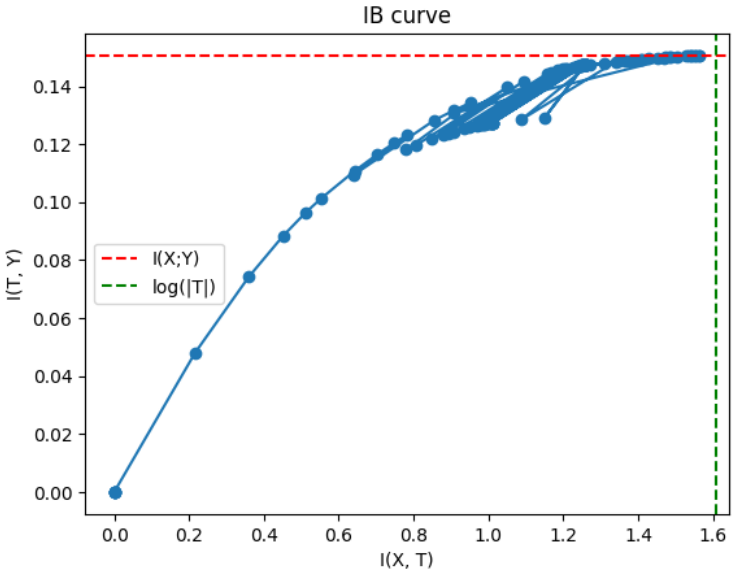

Figure 3: IB curve for a random joint distribution with $|X| = 5$, $|Y| = 3$ and $\beta \in [0; 200]$. We fix the number of iterations for each β to 1000.

As for the DIB curve, the purpose of an IB curve is to illustrate the balance between compression, represented by $I(X;T)$, and prediction, denoted by $I(T;Y)$. Indeed, at the bottom left of the curve, maximizing compression (minimizing $I(X;T)$) prioritizes reducing redundancy and extracting the most essential information from the input data X but we see that by doing this, we lose almost all information about Y and prediction is pretty bad. Conversely, at the other end, maximizing prediction (maximizing $I(T;Y)$) emphasizes capturing as much relevant information as possible to accurately predict the output Y given the latent variable T. By doing this, we see on the curve that we are forced to make little compression.

The bounds represented by a red horizontal line and a green vertical line are the theoretical bounds for $I(T;Y)$ and $I(X;T)$. $I(T;Y)$ is bounded by $I(X;Y)$ because $I(X;Y)$ is the mutual information if we do not compress input data. On the other hand, $I(X;T)$ is bounded by $\log(|T|)$ because

$$I(X;T) = H(T) - H(T \mid X) \leq H(T) \leq \log(|T|)$$

We can prove that $H(T) \leq \log(|T|)$ easily.

Proposition: Let X be a discrete random variable with distribution p. Then $H(X) \leq \log(|X|)$ with equality if and only if p is the uniform distribution.

Proof.

$$
\begin{aligned}
H(X) &= \sum_x p(x) \cdot \log\left(\frac{1}{p(x)}\right) = \mathbb{E}\left[\log\left(\frac{1}{p(X)}\right)\right] \\
&\leq \log\left(\mathbb{E}\left[\frac{1}{p(X)}\right]\right) \quad \text{by Jensen's inequality for concave functions, here } \log(x). \\
&= \log\left(\sum_x p(x) \cdot \frac{1}{p(x)}\right) = \log(|X|)
\end{aligned}
$$

Thus, $H(X) \leq \log(|X|)$. Furthermore, we know that Jensen's inequality[2]is an equality if and only if ϕ is affine or if X is constant. Here, $\phi(x) = \log(x)$ which is clearly not affine so $H(X) = \log(|X|) \iff p(X)$ is constant, i.e. p is the uniform distribution. □

Let's go back to the IB curve. Upon closer inspection, we notice certain points lying below the curve. These points correspond to β values where the algorithm may have struggled to converge, likely due to insufficient iterations. Indeed, setting the number of iterations to 1000 may not always ensure convergence of the algorithm.

Thus, two questions remain:

- What is the optimal number of iterations in order to ensure the convergence of the algorithm?
- Which value of β gives the best tradeoff between compression and prediction? In other words, which β is at the crossroads of the red and the green curves?

2.2.1 Convergence of the algorithm

Instead of fixing the number of iterations, we could choose a metric that ensures convergence. For example, we can compare two consecutive values of the objective function $L_{\text{IB}} = I(X;T) - \beta I(T;Y)$ and stop the algorithm when the difference between these two values is lower than a certain threshold. More precisely, if $L_{\text{IB}}^{(n)}$ is the value of the objective function at the n-th step, we can iterate the algorithm while $|L_{\text{IB}}^{(n)} - L_{\text{IB}}^{(n-1)}| > \theta$ where θ is a chosen threshold. By doing this, we can achieve better results as shown in the plots below:

[2]Jensen's inequality for concave functions: $\mathbb{E}[\phi(X)] \leq \phi(\mathbb{E}[X])$.

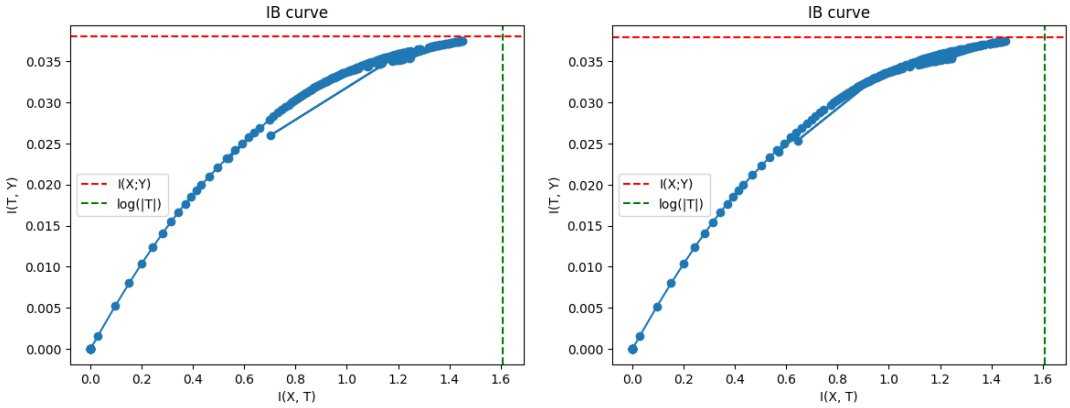

Figure 4: Comparison of the IB curve with a fixed number of iterations for each β (left) and the IB curve using the metric to ensure convergence (right). We fixed the number of iterations to 100 and the threshold θ to 1e-8.

We can clearly see that the algorithm using the metric to ensure convergence has indeed better performance. The curve is smoother and less points are below the curve. We can still see some points under the curve but this comes from the fact that we add a maximum number of iterations in the algorithm to decrease computation time. We fixed this maximum number of iterations to 10'000 which means that we could retrieve the right curve with the first algorithm if we set the number of iterations to 10'000.

2.2.2 Optimal β

The question of determining the ideal β value, where we intersect the red and green curves on the IB curve, is not quite meaningful. It typically leads to a large β value that diminishes compression, essentially negating the utility of the variable T.

Instead, the research of an optimal β relies on the preferred quantity of clusters. The IB curve illustrates the upper bounds of prediction performance achievable when the number of clusters, represented by $I(X;T)$, is fixed. Consequently, this research depends on the specific goals of our analysis.

2.3 General DIB algorithm

Our next objective is to develop the general DIB algorithm. This step is essential as we seek to evaluate and contrast the performance of both IB and DIB algorithms in discrete choice scenarios. To achieve this, we need to have both general algorithms readily available for comparison and analysis.

It appears that the DIB algorithm we developed for geometric clustering serves also as a solution to the general DIB problem. Indeed, our implementation is based on the general DIB algorithm proposed in [2], which accounts for its inherent generality. With this insight, we can seamlessly transition to analyzing the distinctions between IB and DIB algorithms.

3 Discrete choice analysis

3.1 Basics of utility theory

Before diving into the main part of this work where we will apply IB framework to discrete choice, we first recall some basics about utility theory.

Utility theory serves as a fundamental framework in various fields, offering insights into how individuals make decisions amid competing alternatives. Rooted in the concept of rational behavior, utility theory posits that individuals seek to maximize their overall satisfaction when faced with choices. To quantify individuals' satisfaction, utility theory introduces the concept of utility function, a mathematical representation that assigns a numerical value to each possible outcome. These utility functions should reflect individuals' subjective preferences and attitudes toward various options, encapsulating their desires and needs.

In general, a utility function for an alternative i is defined as the sum of two components, a deterministic part and a random component:

$$U_i = V_i + \epsilon_i$$

V_i captures the systematic or predictable aspects of an individual's preferences or the intrinsic value associated with a particular choice. It encompasses factors such as the inherent desirability of an option, its tangible attributes such as cost, and any other deterministic influences that contribute to the individual's overall satisfaction or utility.

The term ϵ_i represents the random component or the stochastic element in the utility function. It embodies the unobservable or unpredictable factors that influence decision-making but cannot be captured by the deterministic component. This component introduces variability into the utility function, accounting for the uncertainty inherent in human decision-making processes.

3.1.1 The logit model

The logit model stands as essential in discrete choice analysis thanks to its simplicity in mathematical formulation, clarity in parameter interpretation, and versatility in accommodating diverse choice scenarios. The model's parameters provide intuitive insights into the relative importance of different attributes or factors influencing choice behavior, facilitating informed decision-making. Furthermore, its flexibility enables its application across various domains, from transportation to economics. Thus, the logit model remains an inevitable tool for analyzing discrete choice behavior, offering a robust and interpretable approach to understanding decision-making processes.

Definition: Let C be a choice set. Assume that the random components of the alternatives ϵ_i are i.i.d. $\mathrm{EV}(\eta, \mu)$. The logit model is derived as

$$P(i \mid C) = \frac{e^{\mu V_i}}{\sum_{j \in C} e^{\mu V_j}}$$

In this context, $V_i = \sum_k \beta_k z_{ik}$, denoting a linear combination of the vector of attributes z_i for alternative i. The β_k coefficients serve as model parameters that necessitate estimation using data. As η does not play any role in the computations, we will fix it to 0.

3.1.2 Nested logit model

The logit model has some limitations. Let's take a basic example to understand that.

Example: Suppose that we want to model the transportation mode choice in a city where the only two choices are car or bus. The basic logit model would be defined as followed:

$$U_{\text{car}} = \beta T + \epsilon_{\text{car}}$$
$$U_{\text{bus}} = \beta T + \epsilon_{\text{bus}}$$

and the choice probability assuming that ϵ_i are i.i.d. $\text{EV}(0, \mu)$ would be:

$$P(\text{car} \mid \{\text{car, bus}\}) = \frac{e^{\mu\beta T}}{e^{\mu\beta T} + e^{\mu\beta T}} = \frac{1}{2}$$
$$P(\text{bus} \mid \{\text{car, bus}\}) = \frac{e^{\mu\beta T}}{e^{\mu\beta T} + e^{\mu\beta T}} = \frac{1}{2}$$

Now suppose that the bus company has two types of buses: red and blue. The new logit model is then:

$$U_{\text{car}} = \beta T + \epsilon_{\text{car}}$$
$$U_{\text{red bus}} = \beta T + \epsilon_{\text{red bus}}$$
$$U_{\text{blue bus}} = \beta T + \epsilon_{\text{blue bus}}$$

and the choice probability with ϵ_i i.i.d. $\text{EV}(0, \mu)$ is:

$$P(\text{car} \mid \{\text{car, blue bus, red bus}\}) = \frac{e^{\mu\beta T}}{e^{\mu\beta T} + e^{\mu\beta T} + e^{\mu\beta T}} = \frac{1}{3}$$
$$P(\text{red bus} \mid \{\text{car, blue bus, red bus}\}) = \frac{e^{\mu\beta T}}{e^{\mu\beta T} + e^{\mu\beta T} + e^{\mu\beta T}} = \frac{1}{3}$$
$$P(\text{blue bus} \mid \{\text{car, blue bus, red bus}\}) = \frac{e^{\mu\beta T}}{e^{\mu\beta T} + e^{\mu\beta T} + e^{\mu\beta T}} = \frac{1}{3}$$

It's evident that there is an issue. In fact, based on our logit model, the car mode share decreases from 50% to 33% when buses are of two different colors. But where does this problem come from?

In the logit model presented earlier, only travel time T is incorporated into the utility functions. This implies that other attributes influencing customer's choice are encompassed by the error term ϵ_i. Among these attributes, some are common to both $\epsilon_{\text{blue bus}}$ and $\epsilon_{\text{red bus}}$, for example fare or comfort. However, this contradicts the logit model's assumption that ϵ_i are independent. Thus, we need to think of a way to overcome this issue.

We have the option to introduce an additional error term into $U_{\text{blue bus}}$ and $U_{\text{red bus}}$ to encompass the

unobserved attributes shared by both alternatives:

$$\begin{aligned} U_{\text{car}} &= \beta T + \epsilon_{\text{car}} \\ U_{\text{red bus}} &= \beta T + \epsilon_{\text{red bus}} + \epsilon_{\text{bus}} \\ U_{\text{blue bus}} &= \beta T + \epsilon_{\text{blue bus}} + \epsilon_{\text{bus}} \end{aligned}$$

With this, model, if we assume that $\epsilon_{\text{blue bus}}$ and $\epsilon_{\text{red bus}}$ are i.i.d. $\text{EV}(0, \mu_b)$, we find that

$$P(\text{blue bus} \mid \{\text{blue bus, red bus}\}) = P(\text{red bus} \mid \{\text{blue bus, red bus}\}) = \frac{e^{\mu_b \beta T}}{e^{\mu_b \beta T} + e^{\mu_b \beta T}} = \frac{1}{2}$$

For the choice between car and bus, we define a new model:

$$\begin{aligned} U_{\text{car}} &= \beta T + \epsilon_{\text{car}} \\ U_{\text{bus}} &= V_{\text{bus}} + \epsilon_{\text{bus}} \end{aligned}$$

where ϵ_{car} and ϵ_{bus} are i.i.d. $\text{EV}(0, \mu)$ and V_{bus} is the expected maximum utility of red bus and blue bus.

Definition: For a set of alternatives C, define

$$U_C \equiv \max_{i \in C} U_i = \max_{i \in C} (V_i + \epsilon_i) \quad \text{and} \quad U_C = V_C + \epsilon_C$$

If ϵ_i are i.i.d. $\text{EV}(0, \mu_b)$, we have

$$V_C = \frac{1}{\mu_b} \ln \left[\sum_{i \in C} e^{\mu_b V_i} \right] \quad \text{and} \quad \mathbb{E}[\epsilon_C] = \frac{\gamma}{\mu_b}$$

With this definition on hand, we can now calculate the probability of choice between car and bus:

$$\begin{aligned} V_{\text{bus}} = \frac{1}{\mu_b} \ln \left(e^{\mu_b V_{\text{blue bus}}} + e^{\mu_b V_{\text{red bus}}} \right) &= \frac{1}{\mu_b} \ln \left(e^{\mu_b \beta T} + e^{\mu_b \beta T} \right) = \beta T + \frac{1}{\mu_b} \ln(2) \\ \implies P(\text{car}) &= \frac{e^{\mu V_{\text{car}}}}{e^{\mu V_{\text{car}}} + e^{\mu V_{\text{bus}}}} \\ &= \frac{e^{\mu \beta T}}{e^{\mu \beta T} + e^{\mu \beta T + \frac{\mu}{\mu_b} \ln(2)}} \\ &= \frac{1}{1 + 2^{\frac{\mu}{\mu_b}}} \end{aligned}$$

Thus, it seems that we solved the problem of the logit model on red/blue buses. The model that we used to overcome the issue of the logit model is called a nested model.

Definition: Let C be the choice set and $C_1, ..., C_M$ a partition of C. The nested logit model is derived as

$$P(i \mid C) = \sum_{i=1}^{M} P(i \mid m, C) P(m \mid C)$$

In the nested model, each alternative i belongs to exactly one nest m which leads to

$$P(i \mid C) = P(i \mid m, C) P(m \mid C)$$

Each nest m is associated with a scale parameter μ_m and an expected maximum utility given by

$$\tilde{V}_m = \frac{1}{\mu_m} \ln \left[\sum_{i \in C_m} e^{\mu_m V_i} \right]$$

Comments on the nested logit model:

1. The ratio μ/μ_m must be estimated from data and needs to be in $[0, 1]$.
2. Going down the nests, μ's must increase and variance must decrease.

With this definition, we can compute the general formulas for the probability of an alternative within its nest and the probability of a particular nest:

$$P(i \mid m) = \frac{e^{\mu_m V_i}}{\sum_{j \in C_m} e^{\mu_m V_j}}$$
$$P(m \mid C) = \frac{e^{\mu \tilde{V}_m}}{\sum_{p=1}^{M} e^{\mu \tilde{V}_p}}$$

In general, μ is normalized to 1 which simplifies the computations of the above probabilities.

3.1.3 Cross-nested logit model

We might consider the following question: what if an alternative could be part of more than one nest? For instance, imagine we have five alternatives: car, bus, train, walking, and cycling, and we wish to establish two nests: motorized and private. In this scenario, car, buse, and train would be in the "motorized" nest, while car, walking, and cycling would be in the "private" nest. However, this situation does not adhere to the structure of a nested logit model, as the alternative "car" is included in both nests. Consequently, we must create a new type of model capable of accommodating this requirement.

Definition: Let C be the choice set and $C_1, ..., C_M$ groups of alternatives which may overlap. Each group is associated with a scale parameter μ_m with the constraint that $\mu/\mu_m \in [0, 1]$ and for each alternative i in nest m we define a degree of membership $\alpha_{im} \in [0, 1]$ with the following constraints:

- Each alternative must belong to at least one nest: for all j, there exists m such that $\alpha_{jm} > 0$
- $\sum_m \alpha_{jm} = 1$ for all j

The cross-nested logit model is derived as

$$P(i \mid C) = \sum_{i=1}^{M} P(i \mid m, C) P(m \mid C)$$

where

$$P(i \mid m, C) = \frac{\alpha_{im}^{\mu_m/\mu} e^{\mu_m V_i}}{\sum_{j \in C} \alpha_{jm}^{\mu_m/\mu} e^{\mu_m V_j}} \quad \text{and} \quad P(m \mid C) = \frac{\left(\sum_{j \in C} \alpha_{jm}^{\mu_m/\mu} e^{\mu_m V_j} \right)^{\frac{\mu}{\mu_m}}}{\sum_{l=1}^{M} \left(\sum_{j \in C} \alpha_{jl}^{\mu_l/\mu} e^{\mu_l V_j} \right)^{\frac{\mu}{\mu_l}}}$$

We can easily see that the cross-nested model is a generalization of the nested model. Indeed, we can define a nested model from this definition by setting $\alpha_{im} = \mathbb{1}_{C_m}(i)$ where $\mathbb{1}_{C_m}(i) = 1$ if $i \in C_m$ and 0 otherwise.

With these different models in mind, we can now dive into the application of information bottleneck methods to the discrete choice.

4 Application of IB/DIB to the discrete choice

The aim of this project is to employ the information bottleneck framework in the context of discrete choice. Specifically, given a discrete choice model like the nested logit model or cross-nested logit model, our objective is to investigate whether it is feasible to reconstruct the nests of the model using the information bottleneck approach applied to the joint distribution $p(x, y)$ derived from the discrete choice model. This tentative seeks to explore the potential of information bottleneck in explaining the underlying structure of discrete choice models, thereby offering insights into the decision-making processes inherent in such models.

4.1 IB/DIB on nested models with synthetic data

We first test the IB framework on nested models with synthetic data. For our first analysis, we take the telephone data available on Biogeme (https://biogeme.epfl.ch/#data). This dataset contains different information about 434 households in Pennsylvania. It involves choices among five calling plans which will be the interesting part for us. Three of the calling plans are flat, which means that customer pays a fixed monthly charge for unlimited calls within a specified geographical area, and the other two are measured, which means that the customer pays a reduced fixed monthly charge for a limited number of calls and additional usage charges for additional calls.

We choose to separate alternatives in two obvious nests : measured and flat. We compute a nested logit model with the following utility functions :

$$
\begin{aligned}
U_1 &= ASC_1 + \beta \cdot \log(\text{cost}_1) + \epsilon_1 + \epsilon_{\text{measured}} \\
U_2 &= \beta \cdot \log(\text{cost}_2) + \epsilon_2 + \epsilon_{\text{measured}} \\
U_3 &= ASC_3 + \beta \cdot \log(\text{cost}_3) + \epsilon_3 + \epsilon_{\text{flat}} \\
U_4 &= ASC_4 + \beta \cdot \log(\text{cost}_4) + \epsilon_4 + \epsilon_{\text{flat}} \\
U_5 &= ASC_5 + \beta \cdot \log(\text{cost}_5) + \epsilon_5 + \epsilon_{\text{flat}}
\end{aligned}
$$

The results of the computation are shown in the table below :

Parameter	**Estimate**	**Robust Asymptotic SE**	**t-statistic**	**p-value**
ASC_1	-0.378246	0.125454	-3.015006	2.723024e-03
ASC_3	0.893446	0.171565	5.207627	2.980374e-07
ASC_4	0.847293	0.393757	2.151815	3.197195e-02
ASC_5	1.405502	0.259374	5.418828	1.004633e-07
BETA_COST	-1.490024	0.252883	-5.892149	7.739943e-09
lambda_measured	0.484798	0.139705	3.470160	5.730726e-04
lambda_flat	0.436216	0.121307	3.595981	3.609402e-04
AIC	960.44			
BIC	988.95			

In this table, $\lambda_{\text{measured}} = 1/\mu_{\text{measured}}$ and $\lambda_{\text{flat}} = 1/\mu_{\text{flat}}$ where μ_{measured} and μ_{flat} are the scale parameters of the two nests.

The model seems satisfying as all the p-values are under 0.05 which means that each parameter is statistically significant and the two λ's satisfy the condition $0 \leq \lambda \leq 1$.

Now that we have a model for the telephone data, our goal is to see if we can retrieve the nests "measured" and "flat" from the IB framework. To do that, we first have to compute the joint distribution $p(x, y) = p(y \mid x)p(x)$ where X is the logarithm of the costs and Y is the five alternatives. We already have $p(y \mid x)$ because we had to use this probability in the likelihood function used to estimate the model. Thus, we only need to find $p(x)$. For that, we compute the empirical distribution for the vectors $(logcost_1, logcost_2, logcost_3, logcost_4, logcost_5)$. For example, if there is only one vector $\vec{v}_1 = (x_1, x_2, x_3, x_4, x_5)$, we set $p(\vec{v}_1) = \frac{1}{434}$ and if we find n identical vectors $\vec{w}_1, ..., \vec{w}_n$, we set $p(\vec{w}_i) = \frac{n}{434}$ $\forall i = 1, ..., n$.

4.2 First tries and analyses [DRAFT]

When we apply the DIB algorithm on telephone data with a NLM, we do not find clear structure. However, by modifying the algorithm in order to allow only 5 clusters at maximum (our goal is to cluster alernatives, not individuals), we find that when we set large betas, i.e. when we want to maximize relevant information, it seems that the choice of the nested logit model is not a good choice. Indeed, the output of the algorithm seems to indicate that we are in presence of a cross-nested logit model where alternative 3 belongs to the two nests as we can see on the tables below:

real choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	22	60	53	0	1
1	15	41	68	0	21
2	2	1	3	3	0
3	33	20	50	0	3
4	1	1	4	0	32

simulated choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	0	0	136	0	0
1	0	82	57	0	6
2	0	0	7	2	0
3	48	3	55	0	0
4	0	0	1	0	37

The left table shows how individuals are distributed across clusters depending on the real choice they made (real data). The right table shows how individuals are distributed across clusters depending on the alternative with maximum probability coming from the model (simulated data).

The table with simulated choice is more interesting for us because it comes directly from the NLM and this is what we are using in the DIB algorithm. In fact, we can see on this table that the algorithm struggles to separate alternative 3 from the other alternatives which seems to indicate the presence of a cross-nested model. Let's see if this is really the case.

We implement a CNLM in order to verify if our guess is correct or not. We set the degrees of membership as $\alpha_{\text{measured}} = \left[1, 1, \frac{1}{2}, 0, 0\right]$ and $\alpha_{\text{flat}} = \left[0, 0, \frac{1}{2}, 1, 1\right]$.

The output of the CNLM is the following :

[I think there is a problem with the λ's as they are greater than 1 but I have the same results than Prof. Bierlaire so does it mean that our model cannot be a model ?]

Parameter	Estimate	Robust Asymptotic SE	t-statistic	p-value
ASC_1	-1.635754	0.496645	-3.293610	1.071442e-03
ASC_3	2.199670	0.556739	3.950990	9.104589e-05
ASC_4	2.426402	1.316582	1.842956	6.602838e-02
ASC_5	4.084728	1.053189	3.878438	1.217014e-04
BETA_COST	-3.354164	0.582177	-5.761416	1.598521e-08
lambda_measured	2.541826	0.624433	4.070612	5.587709e-05
lambda_flat	1.957885	0.714936	2.738544	6.429389e-03
AIC	961.21			
BIC	989.72			

It seems that we did not improve the performance of the model as AIC and BIC are not smaller than before, but we can see some interesting results when we apply again the DIB algorithm to the joint distribution $p(x, y)$ obtained from the CNLM. Indeed, if we set β to a very large number which means that we want to maximize relevant information and give small importance to compression, we obtain the following results :

DIB results for large β : 5 clusters

real choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	21	58	53	0	0
1	11	34	40	0	14
2	3	10	29	3	9
3	37	20	52	0	5
4	1	1	4	0	29

simulated data	**1**	**2**	**3**	**4**	**5**
cluster					
0	0	0	132	0	0
1	0	56	43	0	0
2	0	0	43	2	9
3	21	35	58	0	0
4	0	0	0	0	35

It seems that the DIB algorithm tries to find the clusters depending on the maximal probability between the five alternatives. This probability is directly linked to the CNLM which seems consistent. Furthermore, when we look at the table on the right with maximal probabilities, we see that the algorithm determines five clusters which respect the nests membership. Indeed, the first cluster only has alternative 3, the second cluster has alternatives 2 and 3 which is in the "measured" nest, the third cluster has alternatives 3, 4 and 5 which is in the "flat" nest, etc...

My intuition behind this is that the algorithm cannot recover the exact two nests of "measured" and "flat" because data are too closely linked and there is not enough data for alternative 4. However, the DIB algorithm can gives us "advice" on what model to choose. For example, here with telephone data, it seemed that alternative 3 was not really a "flat" alternative and the DIB algorithm suggests us to try a cross-nested model. Then, by implementing the CNLM, we saw that the results for the model were not really better, but if we applied again the DIB algorithm, the nest structure was respected which was not the case when we applied the algorithm on NLM.

One condition for the DIB to give us "advice" is that we have to take very large β in order to keep the maximum information from data to choose the alternatives. Large β values were a problem on the original IB/DIB, but this is not the case anymore as we set the maximum number of clusters to the number of alternatives and not the number of datapoints.

4.3 Week 6, many tests for different distributions and models

- Change the way I simulate choices. Now I take a random number between 0 and 1 etc...
- Try to take different distributions for $p(cost1, cost2, cost3, cost4, cost5)$. I tried :
  - multinomial lognormal distribution on raw data, logit model
    - Two clusters for $\beta \in [210; 380]$. When DIB finds two clusters, the second cluster has only one element from alternative 5. Surely a very different element compared to the rest of the dataset ?
  - multinomial lognormal distribution without missing data, logit model
    - We have to modify the covariance matrix because it is not positive definite. Two clusters for $\beta \in [11; 16]$. When DIB finds two clusters, the second cluster has only very few elements from alternative 5. DIB seems to find four clusters for many β values but the clustering is not interesting (alternative 4 alone in different clusters, other alternatives in one cluster)
  - multinomial lognormal distribution on raw data but covariance matrix is diagonal, logit model
    - Two clusters for $\beta \in [120; 160]$. When DIB finds two clusters, it identifies alternative 4 as one cluster more or less. DIB seems to find three clusters for many β values but the clustering is not interesting (one cluster has only one individual).
  - multinomial lognormal distribution without missing data but covariance matrix is diagonal, logit model
    - We have to modify the covariance matrix because it is too small for computation. We take the identity matrix. Two clusters for $\beta \in [22; 26]$. When DIB finds two clusters, the second cluster has only very few elements from alternative 5.

  - lognormal distribution for each cost on raw data, nested logit model
    - Two clusters for $\beta \in [5; 6]$. When DIB finds two clusters, it identifies alternative 4 as one cluster more or less. DIB seems to find four clusters for many β values but the clustering is not interesting (alternative 4 in one cluster and the three others are a mix of alternative 1, 2, 3, and 5).
  - lognormal distribution for each cost without missing data, nested logit model
    - Two clusters for $\beta \in [7; 15]$. When DIB finds two clusters, it identify alternative 5 as one cluster more or less.
  - lognormal distribution for cost1,2,3,4 empirical distribution for cost 5, nested logit model
    - Two clusters for $\beta \in [4, 6]$. When DIB finds two clusters, it identifies alternative 4 as a cluster. DIB seems to find four clusters for many β values and when this is the case, one cluster is made of alternative 4,5 and a few 3 which is good news but the three other clusters are a mix of alternative 1,2 and 3.
  - lognormal distribution for cost1,2,3 empirical distribution for cost 4,5, nested logit model
    - Two clusters for $\beta \in [4, 6]$. When DIB finds two clusters, it identifies alternative 4 as a cluster. DIB seems to find four clusters for many β values but the results are not really

interesting.

What I mean by "without missing data" is that I remove the data when the alternative is not available to compute the mean and the covariance.

- multinomial lognormal distribution on raw data, logit model with fixed beta
  - with beta = -1, beta = -1.9 and beta = -3, the clustering identifies alternative 4 as one cluster.
  - with beta = -10, the clustering is awful.

4.4 Week 7, from diagonal covariance matrix to full matrix

Suppose that we take a multivariate lognormal distribution for the distribution of costs, i.e.

$$p(\text{logcost}_1, \text{logcost}_2, \text{logcost}_3, \text{logcost}_4, \text{logcost}_5) \sim \text{Lognormal}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

where μ_i is the mean of $\log(\text{logcost}_i)$ and $\boldsymbol{\Sigma}$ is the covariance matrix of the variables natural logarithm.

We want to see how the DIB algorithm reacts when we play with the covariance matrix of the distribution, i.e. if we assume independence between different costs.

First, suppose that all costs are independent of each other. We take a diagonal matrix where the diagonal elements are the variance of each cost. With this framework, the DIB algorithm cannot really find clusters. In fact, if costs are independent, we would suggest that there is only one cluster because there is no relationship between individuals. This is exactly what is happening. When DIB algorithm finds more than one cluster, there is one big cluster with almost all individuals and the other clusters are only composed of at most five individuals which are probably outliers.

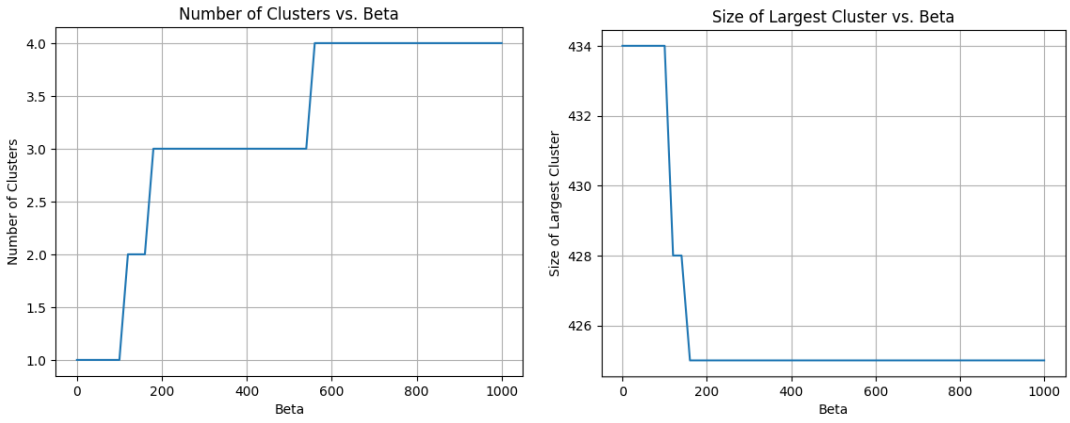

Figure 5: Left: Number of clusters against β when the covariance matrix is diagonal with variances of logcosts as diagonal elements. Right: size of the largest cluster against β. As expected, almost all individuals are in one big cluster and only very few outliers are in a different one.

Let's see what is happening if we take just the identity matrix as covariance matrix.

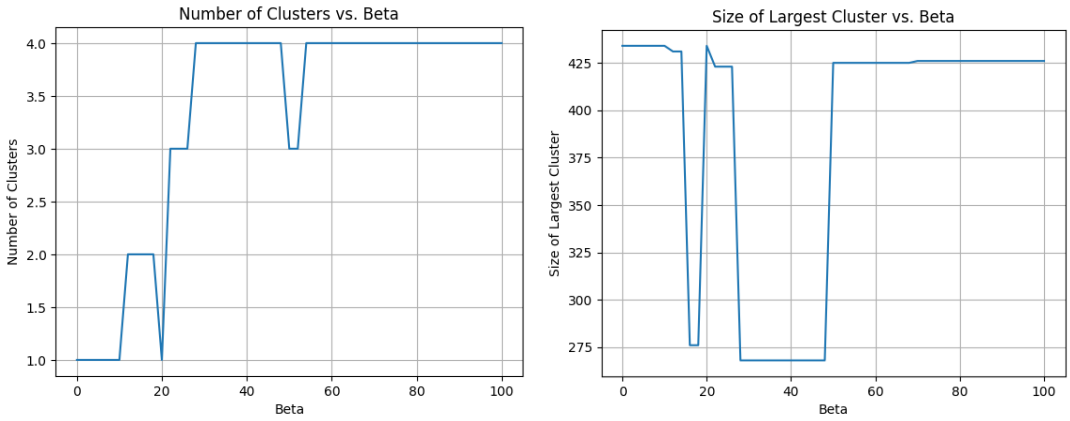

Figure 6: Left: Number of clusters against β when the covariance matrix is the identity. Right: size of the largest cluster against β.

In this case, DIB algorithm only finds a non-trivial clustering for $\beta \in [25; 50]$ where there are two large clusters and two very small where the two large clusters are each composed of every alternatives except alternative 4. It seems that the algorithm tries to "split" the dataset but then when we increase β in order to get more "relevant information" in the clustering, the algorithm do not achieve better result and gives us four clusters with one big cluster and three small ones.

Now, suppose that we add some correlation between cost1 and cost2. Let's see how the DIB algorithm reacts to this change. We take the diagonal matrix with variances as diagonal elements and we add the covariance between logcost_1 and logcost_2 to obtain the new covariance matrix. Results are almost the same as the ones with the diagonal matrix.

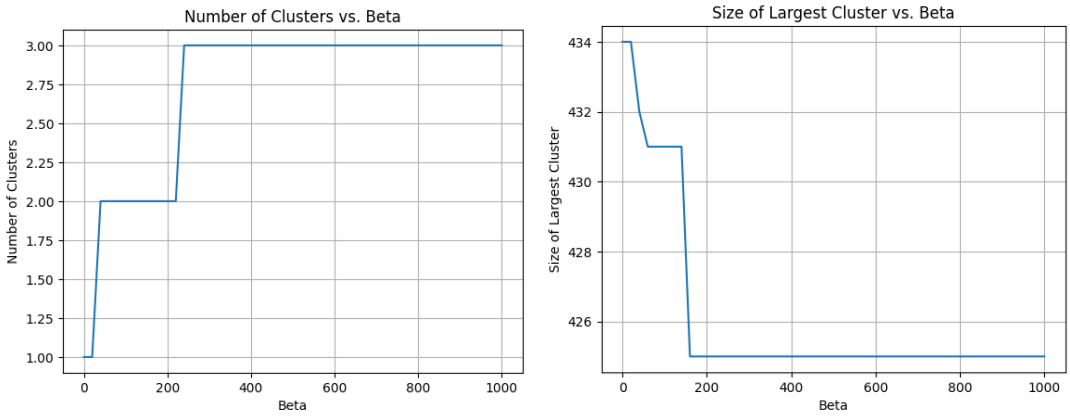

Figure 7: Left: Number of clusters against β when we add covariance between logcost_1 and logcost_2 in the covariance matrix. Right: size of the largest cluster against β.

Let's try to increase the value of covariance between logcost_1 and logcost_2. We cannot really play

with this covariance because if we increase it too much, the determinant of the covariance matrix becomes negative and we cannot compute the distribution. Instead, let's try to take again the identity matrix as the covariance matrix and add some covariance between logcost_1 and logcost_2 into it. We set $cov(\text{logcost}_1, \text{logcost}_2) = 0.9$ to keep the matrix non-singular.

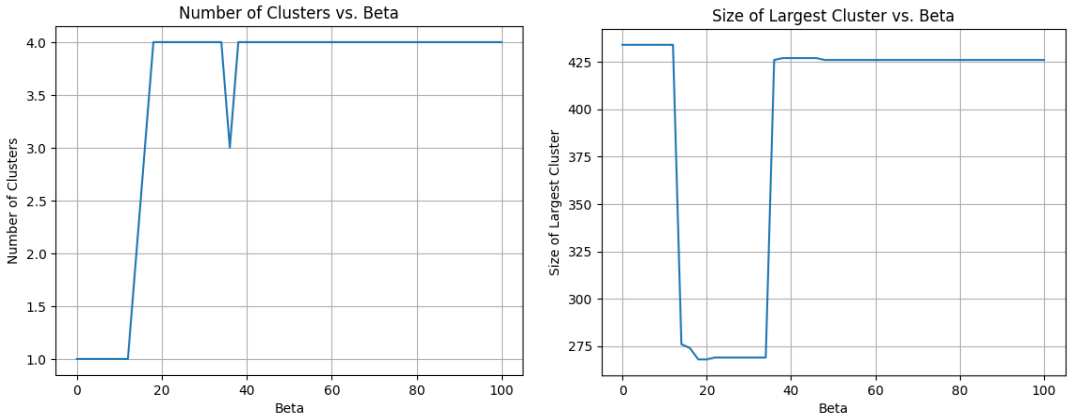

Figure 8: Left: Number of clusters against β when we add covariance between logcost_1 and logcost_2 in the covariance matrix when the latter is the identity matrix. Right: size of the largest cluster against β.

Again, results show one very large cluster and two or three small clusters. It seems that there is almost no difference when we add covariance or not.

Let's see if there is some better clustering when we take the full covariance matrix because if this is not the case, then there is maybe a problem with our approach.

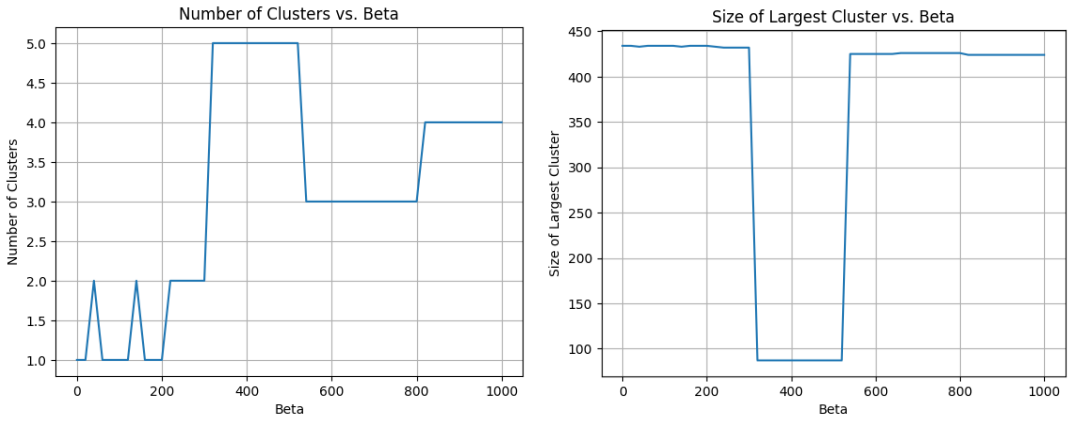

Figure 9: Left: Number of clusters against β when we take the covariance matrix. Right: size of the largest cluster against β.

There is no improvement in the results with the full covariance matrix. Let's try to increase the elements in the covariance matrix.

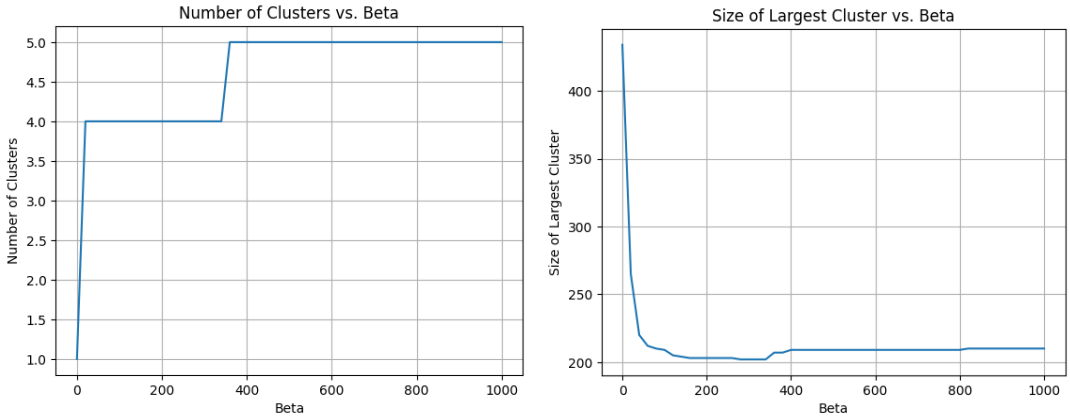

Figure 10: Left: Number of clusters against β when we take the covariance matrix and we multiply it by 100. Right: size of the largest cluster against β.

We finally see something interesting ! When we take the covariance matrix and we multiply it by 100, the DIB algorithm finds some clusters. Thus, we can continue our covariance analysis but we know that we should take big covariances to see an impact on the result.

Let's go back to the diagonal matrix with variances as diagonal elements and covariance between logcost_1 and logcost_2.

Taking the covariance matrix and multiply it by 10 is not enough As we can see below.

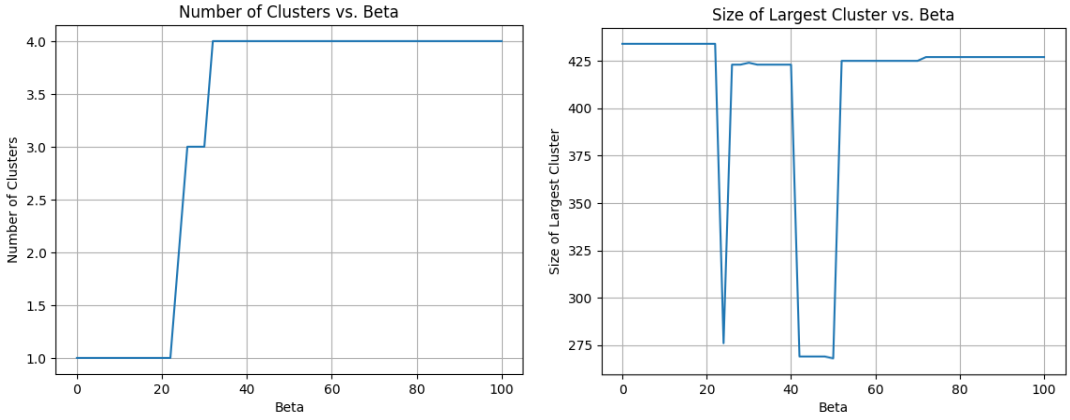

Figure 11: Left: Number of clusters against β when we add covariance between logcost_1 and logcost_2 in the covariance matrix and we multiply it by 10. Right: size of the largest cluster against β.

Indeed, when β increases, the size of the largest cluster does not fall under 425 as in the case where we did not multiply the covariance matrix. We then multiply the covariance matrix by 50 to see what is happening.

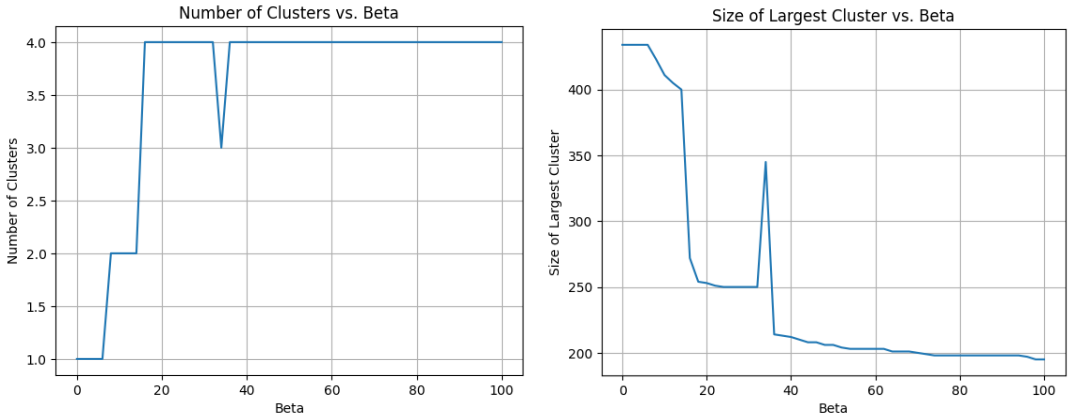

Figure 12: Left: Number of clusters against β when we add covariance between logcost_1 and logcost_2 in the covariance matrix and we multiply it by 50. Right: size of the largest cluster against β.

Something interesting is happening here. It seems that the covariance matrix has finally an impact on the DIB algorithm. Indeed, when β increases, the size of the largest cluster stays at around 200 which indicates a real clustering. This is even more visible if we increase β:

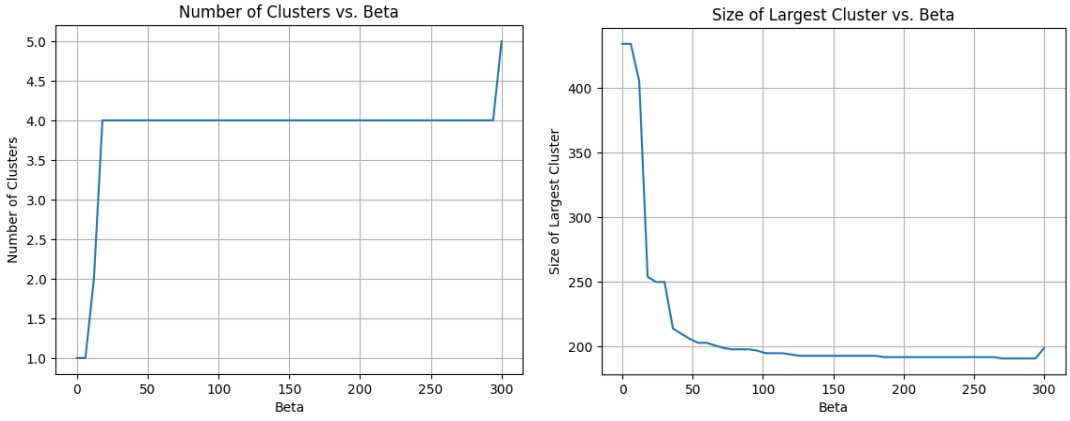

Figure 13: Left: Number of clusters against β when we add covariance between logcost_1 and logcost_2 in the covariance matrix and we multiply it by 50. Right: size of the largest cluster against β.

Let's see what the clustering looks like.

DIB results for $\beta = 200$: 4 clusters

real choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	53	58	72	1	8
1	3	12	42	0	35
2	17	53	63	1	11
3	0	0	1	1	3

max. proba.	**1**	**2**	**3**	**4**	**5**
cluster					
0	49	61	82	0	0
1	0	0	58	0	34
2	0	49	96	0	0
3	0	0	0	2	3

simulated choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	49	74	64	0	5
1	5	9	37	1	40
2	17	52	62	1	13
3	0	0	1	1	3

It seems that the DIB algorithm can find the independence between alternatives 1 and 2 and alternatives

4 and 5 but he cannot separate alternative 3 from them. If we follows the algorithm, alternatives 1,2 and 3 are linked, alternatives 3 and 5 are linked and alternatives 4 and 5 are linked. These huge changes are strange as we only added covariance between logcost_1 and logcost_2. But maybe these results are linked to the value of β. Let's see if we can have only two clusters as expected for smaller values.

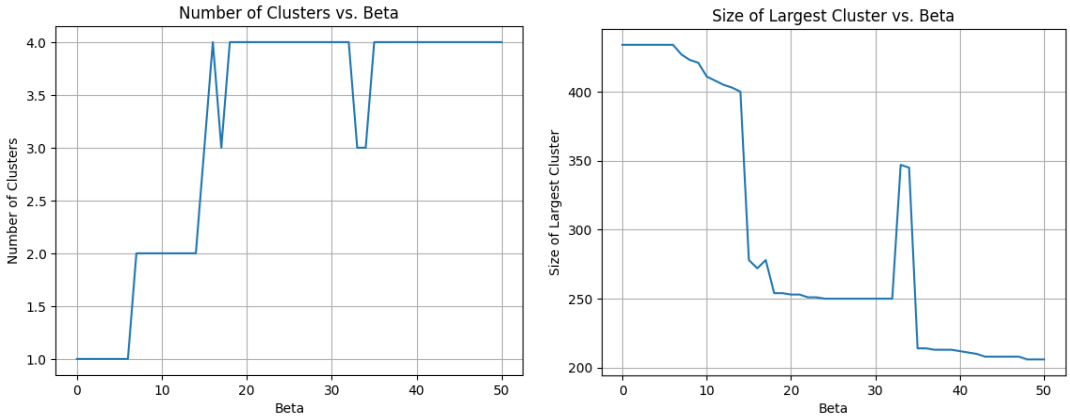

Figure 14: Left: Number of clusters against β when we add covariance between logcost_1 and logcost_2 in the covariance matrix and we multiply it by 50 and we focus on β between 0 and 50. Right: size of the largest cluster against β.

Here, we can see that when β is around 10, the DIB algorithm finds only two clusters as expected. Let's see what the clustering looks like in this case:

DIB results for $\beta = 14$: 2 clusters

real choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	0	1	4	0	29
1	73	122	174	3	28

max. proba.	**1**	**2**	**3**	**4**	**5**
cluster					
0	0	0	1	0	33
1	49	110	235	2	4

simulated choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	0	2	4	0	28
1	71	133	160	3	33

The algorithm manages to isolate alternative 5 in a cluster. So for the algorithm, if we add some correlation between cost 1 and 2, it understands that it has to put these two alternatives in a cluster and it can separate them from alternative 5 but it cannot manage to put them alone in a cluster. This could indicate that alternatives 3 and 4 have a similar behavior to alternatives 1 and 2 whereas alternative 5 is very different from the others.

Now the question is : can the DIB algorithm finds three rational clusters if we add covariance between cost 1 and 3 and cost 2 and 3.

The plots are very similar. Let's see what is the result for $\beta = 14$:

real choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	64	97	111	1	13
1	1	1	4	0	29
2	8	25	63	2	15

max. proba.	**1**	**2**	**3**	**4**	**5**
cluster					
0	49	99	138	0	0
1	0	0	1	0	34
2	0	11	97	2	3

simulated choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	62	115	97	0	12
1	0	2	5	0	29
2	9	18	63	3	20

The algorithm finds three cluster but they are not really meaningful. On the other hand, when we set $\beta = 13$, the algorithm finds almost the same clustering as before with $\beta = 14$ and two clusters. Thus, it seems that adding covariances between costs 1 and 3 and between costs 2 and 3 does not really help the algorithm.

The algorithm seems very sensitive to the value of β. Thus, let's go back to the full covariance matrix to see if we can retrieve a good clustering.

Results with the original covariance matrix multiplied by 50 are very similar to the ones with just covariance between cost 1 and 2. So maybe the algorithm tries to tell us something. Indeed, if the algorithm always tries to make one cluster with alternative 5 and the other cluster with all other alternatives, it is maybe because we can find a better model by taking a nested logit model where the first nest is alternative 5 and the second nest is composed of alternative 1, 2, 3 and 4. Let's see if this is indeed the case.

This is not the case. AIC for logit model is 950 whereas AIC for nested logit model is 970 approximatively. So our intuition is not good.

Another thing that we can compare is how the algorithm reacts when we change the base model. Indeed, we made all the above tests with a multinomial logit model for the distribution of $p(y \mid x)$. Let's see what are the differences when the model is a nested logit model where the first nest is "measured alternative" and the second nest is "flat alternative". We remember the results for multinomial logit model with full covariance matrix multiplied by 50:

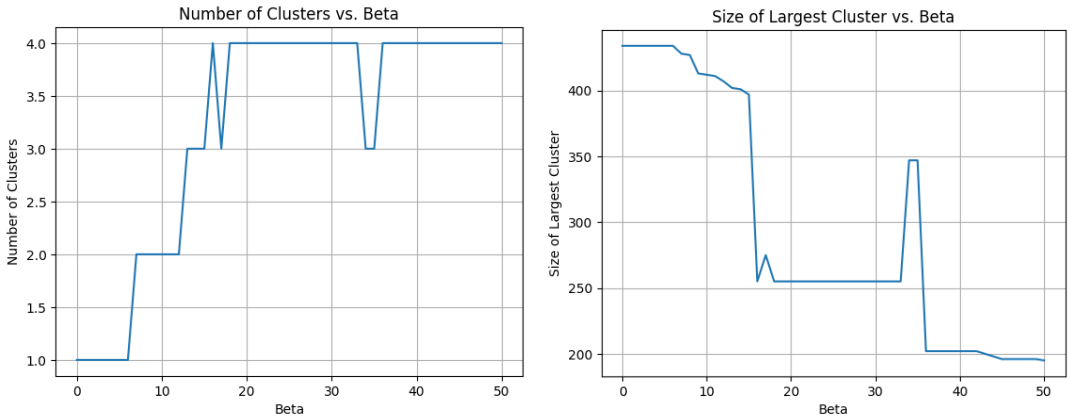

Figure 15: Left: Number of clusters against β with full covariance matrix multiplied by 50 and we focus on β between 0 and 50. Right: size of the largest cluster against β.

Now, we change the base model to a nested logit model as said before.

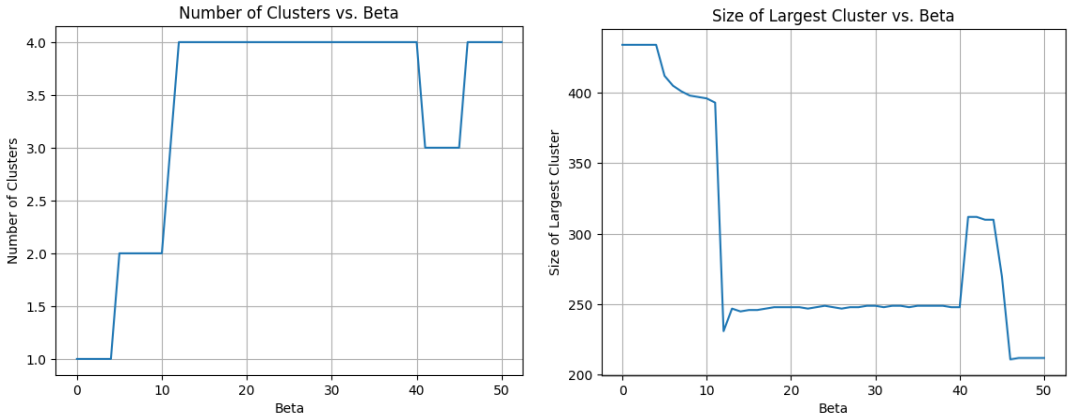

Figure 16: Left: Number of clusters against β with full covariance matrix multiplied by 50 and we focus on β between 0 and 50. Right: size of the largest cluster against β.

There is almost no difference but we can see that the algorithm finds three clusters for more β values when we consider NLM rather than a LM and for these values, the size of the largest cluster is smaller. Let's see what are the differences between these clusterings with three clusters.

DIB results for $\beta = 34$ with LM : 3 clusters

real choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	70	116	140	2	19
1	3	7	37	0	31
2	0	0	1	1	7

max. proba.	**1**	**2**	**3**	**4**	**5**
cluster					
0	49	110	188	0	0
1	0	0	48	0	30
2	0	0	0	2	7

simulated choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	67	128	130	1	21
1	4	7	33	1	33
2	0	0	1	1	7

DIB results for $\beta = 45$ with NLM : 3 clusters

real choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	41	27	56	2	3
1	32	95	117	0	26
2	0	1	5	1	28

max. proba.	**1**	**2**	**3**	**4**	**5**
cluster					
0	48	8	73	0	0
1	0	77	183	0	10
2	0	0	0	2	33

simulated choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	41	39	46	0	3
1	34	87	124	1	24
2	0	2	3	1	29

With NLM, DIB algorithm seems to understand that alternatives 4 and 5 are in the same cluster but he cannot separate alternative 3 from alternatives 1 and 2. Furthermore, DIB with LM manages to separate alternative 2 from alternative 5 which is not the case with NLM.

4.5 THEORY

When we apply DIB algorithm to a LM with a multivariate log-normal distribution with full covariance matrix for the distribution of costs, it tells us that we should consider a cross-nested logit model. Why a cross-nested and not a nested model which is the one who gives the best AIC and BIC ?

I think this is because DIB algorithm does not take into account model complexity. Rather, it favors the best fit. Indeed, I made a test. I produce 1000 simulated choices from the LM model. Then I

produce the NLM and the CNLM and I compute the log-likelihood for these 1000 simulated choices for the two models. In 80% of the cases, the CNLM has a bigger LL than the NLM. This is why the DIB pushes for a CNLM instead of a NLM.

4.6 Week 8

4.6.1 Step-by-step guide for DIB algorithm

1) We consider a dataset of individuals with some attributes and different alternatives. We choose some interesting attributes as X and we consider alternatives as Y.

2) Our goal is to have $p(x, y)$ in order to apply DIB algorithm. Thus, our strategy is to compute $p(x)$ and $p(y \mid x)$ and multiply them to obtain what we want.

3) To compute $p(x)$, we choose a distribution that fits well our data (normal distribution, log-normal distribution, ...) and we compute for each individual the value of this distribution. For this step, I use the function "multivariate_lognormal_pdf".

4) To compute $p(y \mid x)$, we choose a model which seems appropriate to our problem (logit, nested logit, ...) and we compute for each individual the probability of each alternative to be chosen given the attributes of the individual. For this step, I use the function "log_likelihood_telephone_LM" to define the likelihood function and "estimate_nested_logit" to compute the model's coefficients.

5) We multiply $p(x)$ and $p(y \mid x)$ for each individual to find the joint distribution $p(x, y)$.

6) We can now apply DIB algorithm to this joint distribution and see what clusters the algorithm finds for different values of β. For this step, I use the function "geom_DIB_on_alternatives". The name is wrong, this is not for geometric clustering, this is just the DIB algorithm.

4.6.2 Change the distribution of $p(x)$ by handling missing values

Until now, we did not really consider missing values but this could be one of the reason why the algorithm clustering is not very good. Thus, we will analyze what happens when we handle missing values for alternatives 4 and 5 (there is no missing values for alternatives 1, 2 and 3).

We take again a multivariate lognormal distribution for $p(x)$ but this time, we compute the mean vector differently. Instead of just taking the mean of the logarithm of each logcost (when we write $Y \sim Lognormal(\mu_Y, \sigma_Y^2)$, μ_Y and σ_Y^2 are the mean and the standard deviation of $\log(Y)$), we change the procedure for logcost_4 and logocost_5. We remove individuals where alternatives are not available and we take the mean over individuals who could choose alternatives 4 or 5.

For the covariance matrix, we compute the covariances by removing missing values and we multiply the variance for logcost_4 by 5 because if we don't do it, the matrix has a negative determinant.

The results are quite different to the ones where we did not take into account missing values.

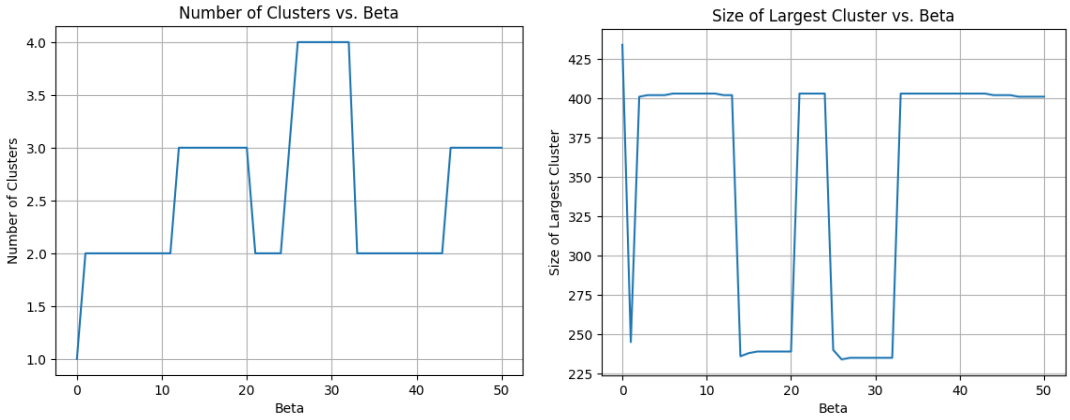

Figure 17: Left: Number of clusters against β with full covariance matrix multiplied by 50 when we take into account missing values. Right: size of the largest cluster against β.

It seems that there are two big cases : when the largest cluster has around 230 elements and when the largest cluster has around 400 elements. Let's see what these two cases of clustering look like:

DIB results for $\beta = 20$: 2 clusters

real choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	72	122	174	3	32
1	1	1	4	0	25

max. proba.	**1**	**2**	**3**	**4**	**5**
cluster					
0	49	110	236	2	6
1	0	0	0	0	31

simulated choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	71	133	161	3	35
1	0	2	3	0	26

DIB results for $\beta = 30$: 4 clusters

real choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	35	63	57	2	8
1	1	1	3	1	5
2	36	58	114	0	27
3	1	1	4	0	25

max. proba.	**1**	**2**	**3**	**4**	**5**
cluster					
0	5	0	152	0	0
1	0	0	4	2	5
2	44	110	80	0	0
3	0	0	0	0	31

simulated choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	20	55	82	0	0
1	0	2	3	3	3
2	51	76	76	0	32
3	0	2	3	0	26

DIB results for $\beta = 40$: 2 clusters

real choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	72	122	174	3	32
1	1	1	4	0	25

max. proba.	**1**	**2**	**3**	**4**	**5**
cluster					
0	49	110	236	2	6
1	0	0	0	0	31

simulated choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	71	133	161	3	35
1	0	2	3	0	26

DIB results for $\beta = 50$: 3 clusters

real choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	0	0	1	0	2
1	72	122	174	3	30
2	1	1	3	0	25

max. proba.	**1**	**2**	**3**	**4**	**5**
cluster					
0	0	0	0	0	3
1	49	110	236	2	4
2	0	0	0	0	30

simulated choice	**1**	**2**	**3**	**4**	**5**
cluster					
0	0	0	1	0	2
1	71	133	161	3	33
2	0	2	2	0	26

It seems that the algorithm can always isolate alternative 5 from the other alternatives but it cannot really separate alternatives 1, 2, 3 and 4.

4.7 New step-by-step guide

1) We create a model (here a logit model). We estimate the model on the original telephone data.

2) We compute the mean μ and the covariance matrix Σ of the original telephone data (+ we handle missing values). From that, we generate 1000 samples from a multivariate lognormal distribution of mean μ and covariance matrix Σ.

3) From the simulated data, we discretize the multivariate lognormal distribution (so we recompute the mean and the covariance matrix from the simulated data in order to avoid non-positive-definite covariance matrix).

4) We associate a probability $p(x)$ to each of the 1000 simulated data. This probability comes from the discretization that we just made.

5) We simulate utilities by using the estimates of the model in step (1) and by adding an error term $\epsilon_i \sim \mathrm{EV}(0, 1)$. From these utilities, we can compute $p(y \mid x)$ for each alternative and then simulate a choice with Evangelos procedure.

6) We can now compute $p(x, y)$ and apply DIB algorithm on it.

5 FOR YOU AND ME

5.1 TO DO LIST

- Oral presentation -> What we understand, what we don't understand and what we will do next. The goal is to hear from Michel what we can do now.
- Try to make a better multivariate lognormal distribution by handling the missing values TUESDAY
- Explain step by step how I can make a plot (which functions I use etc...) TUESDAY

5.2 MINIMAL GOAL

- Show that we understand how the IB/DIB works and how it is relevant for the discrete choice model
- The goal of applying IB/DIB to discrete choice is to justify the choice of our model (nested, cross-nested, etc...)

5.3 Meeting

5.4 Questions

- Does it really make sense to take a very big covariance matrix ? Because the diagonal elements should represent the variance of the logcosts but then if I multiply the covariance matrix by a large number then the diagonal elements are not at all the variances.

6 References in APA format

[1] Tishby, N., Pereira, F. C., & Bialek, W. (2000). The information bottleneck method. arXiv preprint physics/0004057.

[2] Strouse, D. J., & Schwab, D. J. (2017). The deterministic information bottleneck. Neural computation, 29(6), 1611-1630.

[3] Strouse, D. J., & Schwab, D. J. (2019). The information bottleneck and geometric clustering. Neural computation, 31(3), 596-612.

[4] MacKay, D. J. (2003). Information theory, inference and learning algorithms. Cambridge university press.

[5] Ben-Akiva, M. E., & Lerman, S. R. (1985). Discrete choice analysis: theory and application to travel demand (Vol. 9). MIT press.