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:

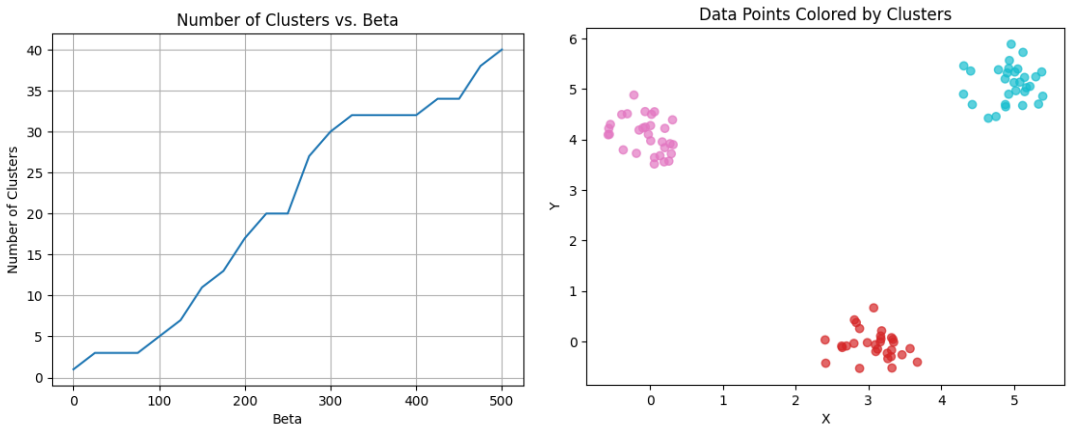

Figure 1: I WILL RECOMPUTE THE FIRST IMAGE WITH MORE BETA VALUES Left: Number of clusters determined by the DIB algorithm against β for 50 iterations. Right: Data points colored by clusters using the DIB algorithm with $|X| = 90$, $\beta = 3.7$ and 100 iterations.

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.

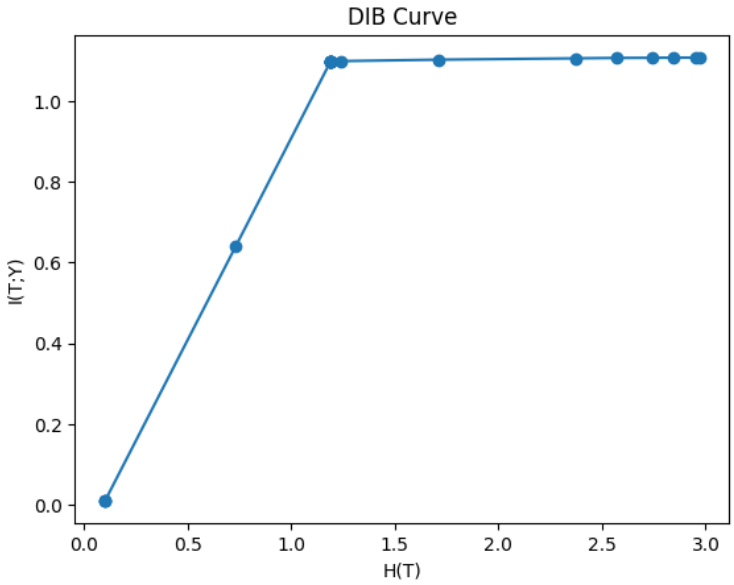

Figure 2: DIB curve for the multivariate gaussian dataset. β takes its values in a uniform vector of 11 points between 0 and 1 and a uniform vector of 9 points between 4 and 500. We fix the number of iterations for each β to 50.

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:

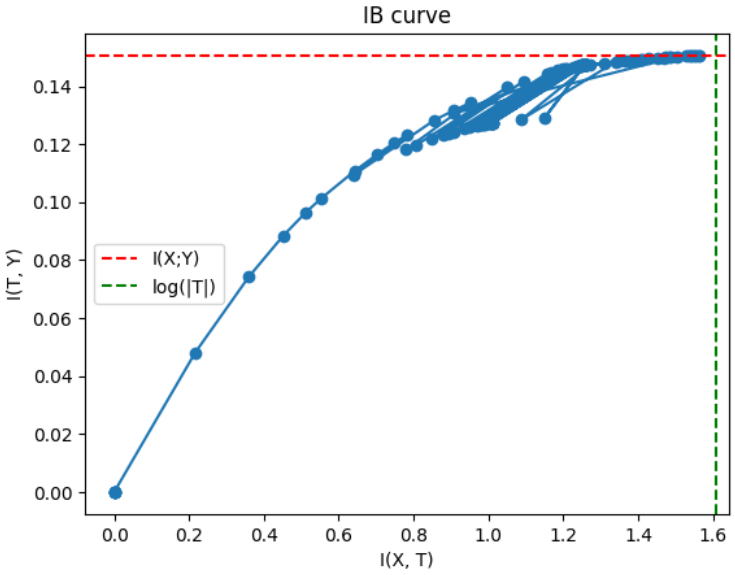

Figure 3: IB curve for a random joint distribution with $|X| = 5$, $|Y| = 3$ and $\beta \in [0; 200]$. We fix the number of iterations for each β to 1000.

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 lines 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: $\mathbb{E}[\phi(X)] \leq \phi(\mathbb{E}[X])$.

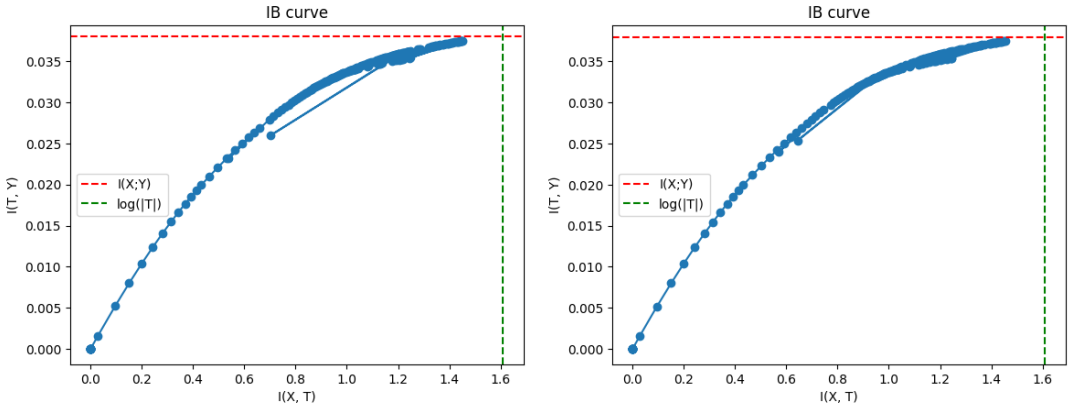

Figure 4: Comparison of the IB curve with a fixed number of iterations for each β (left) and the IB curve using the metric to ensure convergence (right). We fixed the number of iterations to 100 and the threshold θ to 1e-8.

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 $\alpha_{jm} > 0$
- $\sum_m \alpha_j m = 1$ 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 $p(x)$.

5 FOR YOU AND ME

5.1 TO DO LIST

5.2 MINIMAL GOAL

- Show that we understand how the IB/DIB works and how it is relevant for the discrete choice model

5.3 What I did since last Friday

- I implemented a NLM for telephone data based on the R file from Evangelos where he implemented a NLM for Swissmetro. My NLM on telephone data has the same outputs than if I run the code of Professor Bierlaire -> my code should be correct.
- I changed my function information_bottleneck_convergence by using the objective function to check convergence instead of the Euclidean norm on $q(t)$.
- I made the first tests of IB/DIB on my NLM model for telephone data. For $p(y \mid x)$, I used the probability that we find in the computation of NLM (P_1, P_2, P_3, P_4 and P_5) and for $p(x)$, I made a Monte Carlo simulation on the sum of the logcost. I don't know if this was a good idea but for me, only people with exactly same alternatives will have the same sum of logcost so I thought that it could be relevant to use this. I tried different nests : measured vs flat on complete data, measured vs flat without alt. 4 on a subset, alt. 1,2,3 vs alt. 4,5 on a subset, alt. 1,2,3 vs alt. 5 on a subset and alt. 1,2 vs alt. 3,5 on a subset. I did not find any good results I think.
- I wrote a new chapter in my report "discrete choice analysis" where I just talk about the basics of LM, NLM and CNLM.
- Update of the README.

5.4 Questions

- What is the goal of applying IB/DIB to discrete choice ? Because, to apply IB, we need to have p(x,y) which comes from the discrete choice model (nested, cross-nested, etc...) so is the goal just to see if we can have the same clusters/nests with IB and NLM ? Or am I missing something ?

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.