Modelling in Neuroscience

Nicolás Violante

MVA 2021-2022

1 Introduction

The aim of this project is to analyze the Information Bottleneck method [4], a compression procedure based on Information Theory principles. A particular case of interest is the so-called *hard-cluster*, we implement an iterative procedure for this case, the Agglomerative Information Bottleneck [2], and apply it to the task of word clustering [3].

This report is divided as follows. We first define some important quantities that will be used throughout the report. Then we introduce the Information Bottleneck method and its *hard-cluster* limit. Then we present the Agglomerative Information Bottleneck algorithm. Finally we describe the experimental setup and show how can we use the algorithm for word clustering.

1.1 Information Theory Background

Let X be a discrete random variable, the Shannon entropy is a measure of how uncertain is the outcome of this random variable.

$$H(X) = -\sum_x p(x) \log p(x)$$

Likewise, we define the conditional entropy of X given Y as the average Shannon entropy for $X|Y$, averaged over all possible values of Y

$$\begin{aligned} H(X|Y) &= \sum_y p(y) \left(-\sum_x p(x|y) \log p(x|y) \right) \\ &= -\sum_{x,y} p(x,y) \log p(x|y) \end{aligned}$$

Another quantity of interest for us is the mutual information between X and Y. It's the difference between the uncertainty of X and the uncertainty of X when we know Y, therefore it is a measure of how much information about X is contained by Y.

$$\begin{aligned} I(X,Y) &= H(X) - H(X|Y) \\ &= \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \end{aligned}$$

Note that the mutual information corresponds to the Kullback-Leibler divergence between $p(x, y)$ and the product measure $p(x)p(y)$, which we denote $\mathrm{KL}(p(x, y)||p(x)p(y))$. The KL divergence is a positive similarity measure between probability distributions, however it is not a distance because its not symmetric. The mutual information $I(X, Y) = 0$ when the X and Y are independent, a totally reasonable result that matches our intuition about the meaning of independence: knowing X does not tells us nothing about Y.

Finally, another measure of similarity between probability distributions is the Jensen-Shannon divergence. It is defined in terms of the KL divergence and its main advantage is that, unlike KL, it is symmetric.

$$\mathrm{JS}(p(x)||q(x)) = \frac{\mathrm{KL}(p(x)||q(x)) + \mathrm{KL}(p(x)||q(x))}{2}$$

The JS diverenge plays a major rol in the Agglomerative Information Bottleneck algorithm

1.2 The Information Bottleneck

Let X and Y be two random variables. Given their joint distribution $p(x, y)$, we want to find a compact representation $\tilde{X}$ of the variable X that retains as much meaningful information about Y as possible. Thus, we need to find a stochastic mapping $p(\tilde{x}|x)$ such that

$$\min_{p(\tilde{x}|x)} I(X, \tilde{X}) - \beta I(\tilde{X}, Y) \tag{1}$$

The parameter $\beta > 0$ controls the trade-off between compression and retained predictive power about Y. The method is called Information Bottleneck because we are indeed squeezing the information of X through a bottleneck given by $\tilde{X}$ before going to Y. This is behaviour is made clear with the Markov chain of Equation 2. We encode the relevant information that X contains about Y in the new compressed representation $\tilde{X}$.

$$X \to \tilde{X} \to Y \tag{2}$$

One interesting aspect of this formulation as a functional minimization problem is that it yields a set of self-consistent equations

$$\begin{cases} p(\tilde{x}|x) = \frac{1}{Z(\beta,x)} p(\tilde{x}) e^{-\beta \mathrm{KL}(p(y|x)||p(y|\tilde{x}))} \\ p(\tilde{x}) = \sum_x p(\tilde{x}|x)p(x) \\ p(y|\tilde{x}) = \sum_x p(\tilde{x}|x)p(y|x)\frac{p(x)}{p(\tilde{x})} \end{cases} \tag{3}$$

However, this is just a formal solution because the terms $p(\tilde{x}|x)$ and $p(y|\tilde{x})$ appear in each other's respective equations. Nevertheless, this set of equations can be solved iteratively with convergence guaranties [4]. A nice property of this formulation is that the KL divergence naturally emerges as the measure of similarity between $p(y|x)$ and $p(y|\tilde{x})$.

1.3 Hard Clustering Limit

For $\tilde{X}$ with finite cardinality m, the limit case $\beta \to \infty$ the self-consistent equations (1) correspond to a hard partition of X into m clusters. Let's examine that in detail.

In the limit the self-consistent equations become

$$\begin{cases} p(\tilde{x}|x) = \begin{cases} 1 \text{ if } x \in \tilde{x} \\ 0 \text{ otherwise} \end{cases} \\ p(\tilde{x}) = \sum_x p(\tilde{x}, x) \\ p(y|\tilde{x}) = \sum_x p(\tilde{x}|x) p(y|x) \frac{p(x)}{p(\tilde{x})} \end{cases} \tag{4}$$

The first equation for $p(\tilde{x}|x)$ in Equations 4 states that the we have indeed hard clusters. The second equation remains unchanged. Finally, we can rewrite the third equation as $p(y, \tilde{x}) = \sum_{x \in \tilde{x}} p(y, x)$. In the next section we will see how to use this equation for the updates of algorithm.

Note that in this scenario we only care about maximizing $I(Y, \tilde{X})$. Since we will start from a trivial partition of singletons, one for each element of X, this means that we seek the partition into m disjoint clusters that results in the smallest loss of mutual information. The different "components" of X are simple grouped together according to the mutual information with respect to Y.

1.4 Agglomerative Information Bottleneck algorithm

Let $N = |X|$ be the number of possible values that X can take. The algorithm starts with the trivial partition of X into N singletons, one for each possible value. At each step, we merge together two components of the current partition. These two components are selected by locally minimizing the loss of mutual information. At the last iteration, the final partition has one single component consisting of all possible values of X. By this point, iterated over all possible sizes of partitions, from N to 1.

We now explain the merging procedure. Let $Z_m = \{z_1, \dots z_m\}$ be the current partition, with m elements. The next partition is denoted Z_{m-1}. The only difference between them is that two components $z_i, z_j \in Z_m$ had been merged together into a component $\bar{z}$. For this new component we define

$$\begin{cases} p(\bar{z}) = p(z_i) + p(z_j) \\ p(y|\bar{z}) = \frac{1}{p(\bar{z})} \left(p(z_i, y) + p(z_j, y) \right) \\ p(\bar{z}|x) = \begin{cases} 1 \text{ if } x \in z_i \text{ or } x \in z_j \\ 0 \text{ otherwise} \end{cases} \end{cases} \tag{5}$$

This equations verify the hard-clustering limit conditions of Equation 4. Now we define two quantities to measure the quality of the merge

- Y-decrease information: $\delta I_Y(z_i, z_j) = I(Z_m, Y) - I(Z_{m-1}, Y)$
- X-decrease information: $\delta I_X(z_i, z_j) = I(Z_m, X) - I(Z_{m-1}, X)$

Since our goal is to find the merge that locally minimizes the loss of information, we need to find z_i and z_j with minimal Y-decrease information. It can be shown that this quantity can be efficiently computed as

$$\delta I_Y(z_i, z_j) = (p(z_i) + p(z_j))\mathrm{JS}(p(y|z_i), p(y|z_j)) \tag{6}$$

where $\mathrm{JS}(p(y|z_i), p(y|z_j))$ is the Jensen-Shannon divergence between $p(y|z_i)$ and $p(y|z_j)$. Equation 6 yields a nice and intuitive interpretation on the loss of mutual information: it is the the *weight* of the components $p(z_i) + p(z_j)$ times the *distance* $\mathrm{JS}(p(y|z_i), p(y|z_j))$.

Therefore, in each step, we compute the JS-divergence between all pairs of components z_i, z_j and merge those with the smallest Y-information decrease.

The pseudo-code for the full algorithm is given in Figure 1

Input: Empirical probability matrix $p(x, y)$, $N = |X|$, $M = |Y|$

Output: Z_m : *m-partition* of X into m clusters, for every $1 \leq m \leq N$

Initialization:

- Construct $Z \equiv X$
  - For $i = 1...N$
    - $z_i = \{x_i\}$
    - $p(z_i) = p(x_i)$
    - $p(y|z_i) = p(y|x_i)$ for every $y \in Y$
    - $p(z|x_j) = 1$ if $j = i$ and 0 otherwise
  - $Z = \{z_1, ..., z_N\}$
- for every $i, j = 1...N$, $i < j$, calculate
  $d_{i,j} = (p(z_i) + p(z_j)) \cdot JS_{\Pi_2}[p(y|z_i), p(y|z_j)]$
  (every $d_{i,j}$ points to the corresponding couple in Z)

Loop:

- For $t = 1...(N-1)$
  - Find $\{\alpha, \beta\} = argmin_{i,j}\{d_{i,j}\}$
    (if there are several minima choose arbitrarily between them)
  - Merge $\{z_\alpha, z_\beta\} \Rightarrow \bar{z}$:
    - $p(\bar{z}) = p(z_\alpha) + p(z_\beta)$
    - $p(y|\bar{z}) = \frac{1}{p(\bar{z})}(p(z_\alpha, y) + p(z_\beta, y))$ for every $y \in Y$
    - $p(\bar{z}|x) = 1$ if $x \in z_\alpha \cup z_\beta$ and 0 otherwise, for every $x \in X$
  - Update $Z = \{Z - \{z_\alpha, z_\beta\}\} \bigcup \{\bar{z}\}$
    (Z is now a new $(N-t)$-*partition* of X with $N - t$ clusters)
  - Update $d_{i,j}$ costs and pointers w.r.t. $\bar{z}$
    (only for couples contained z_α or z_β).
- **End For**

Figure 1: Agglomerative Information Bottleneck algorithm [2]

2 Experiments

This section describes the experiments performed. First we describe the usage of the Agglomerative Information Bottleneck in hand-made example with known probability distribution. With this small example we will illustrate the most important components of the algorithm. The second experiment consists in an empirical estimation of the distribution of words in text data.

2.1 Example

For this example, we use a hand-made joint probability matrix to illustrate the method step by step and become familiar with it. Consider two random variables $X = \{0, 1, 2, 3, 4\}$ and $Y = \{0, 1\}$ with the following joint distribution

	Y=0	**Y=1**
X=0	0.16	0.04
X=1	0.255	0.045
X=2	0.27	0.03
X=3	0.02	0.08
X=4	0.03	0.07

We can compute the starting mutual information $I(X, Y) = I(Z_5, Y) = 0.1426$. Here $Z_5 = \{\{0\}, \{1\}, \{2\}, \{3\}, \{4\}\}$ denotes the initial partition of singletons. Let's walk trough the first steps of the algorithm

1. In the first iteration, the components $z_0 = \{0\}$ and $z_1 = \{1\}$ are selected to be merged. This incurs in a loss of information of $\delta I_Y(z_0, z_1) = 0.0010853$, which amounts for 0.76% of our original information $I(Z_5, Y)$. After this first step we end up with a partition $Z_4 = \{\{2\}, \{3\}, \{4\}, \{0, 1\}\}$

2. In the second iteration, the selected components to merge are $z_2 = \{4\}$ and $z_3 = \{0, 1\}$. In this case the loss of information is $\delta I_Y(z_2, z_3) = 0.001340$ and the resulting partition is $Z_3 = \{\{2\}, \{3\}, \{4, 0, 1\}\}$

3. For the third iteration we merge $z_0 = \{2\}$ and $z_1 = \{3\}$ and obtain the partition $Z_2 = \{\{2, 3\}, \{4, 0, 1\}\}$. The loss of mutual information is $\delta I_Y(z_0, z_1) = 0.004238$

4. The algorithm terminates with the single partition $Z_1 = \{0, 1, 2, 3, 4\}$

2.2 Word clustering in documents

An interesting application of the method consists in clustering words in different kinds of texts. In this context, the random variable X corresponds to the different words, and Y to the topic of the text. For the estimation of the joint distribution $p(x, y)$ we can compute the frequency of occurrence of each word in the text. We expect that some key words are more frequent than others depending on the topic of the text. Then, using the Agglomerative Information Bottleneck, we can group several words together according to their mutual information with respect to the topic.

Dataset We use the *20Newsgroup* dataset. It consists of about 18000 online discussion posts about several topics in English. We focus on only two

categories: science (cryptography, electronics, and space) and politics (guns, Middle-East, and miscellaneous). There roughly 1500 posts for each one of these two categories. This means that approximately we should be working with balanced dataset. Later on, after the joint distribution estimation, we will explicitly verify this. Having a balanced dataset is important because we want a comparable number of words for each topic.

Pre-processing Since this conversations were originally published in an online chat, they contain text metadata that can bias the word-counting. Thus, we need to remove the headers, footers, and quotes from the posts. This way we make sure to count only words related to the topics of conversation. We also lowercase all the words and remove punctuation marks and special characters. Moreover, since for our experiment we are interested in meaningful words, we restrict ourselves only to noun. This is an easy way to avoid filler and common words like *the* or *and*.

Lemmatization We substitute each word for its lemma. For example, the words *is*, *was*, and *are* have the same lemma *be*. The word *better* has *good* as its lemma. Since this is a rather complicated aspect of text processing that requires expertise in English linguistics, we use the lemmatization models available in *spaCy* [1].

Once we have our pre-processed and lemmatized text, the count the occurrence of each word. For the purpose of illustration, we only keep the most frequent 10 words of each topic. Since there are some overlapping of those words, we end up with 17 words: *information*, *year*, *chip*, *number*, *fact*, *people*, *right*, *country*, *time*, *encryption*, *weapon*, *government*, *thing*, *system*, *datum*, *state*, and *space*.

After normalizing by the total number of words we finally have our empirical estimation of the joint distribution $p(x, y)$. We can verify that we have indeed a balanced dataset by checking that $p(y = \text{science}) = 0.495$.

Finally we apply Agglomerative Information Bottleneck to the obtained $p(x, y)$. This returns all possible partitions, from cardinality 17 to 1. We show some results in Table 1.

['information'], ['year'], ['chip'], ['number'], ['fact'], ['people'], ['right'], ['country'], ['time'], ['encryption'], ['weapon'], ['government'], ['thing'], ['system'], ['datum'], ['state'], ['space']
['information'], ['chip'], ['fact'], ['right'], ['encryption'], ['weapon'], ['government'], ['space'], ['year', 'thing'], ['people', 'time'], ['number', 'system', 'datum', 'country', 'state']
['chip'], ['right'], ['government'], ['space'], ['year', 'thing'], ['fact', 'people', 'time'], ['encryption', 'weapon', 'information', 'number', 'system', 'datum', 'country', 'state']
['space'], ['government', 'encryption', 'weapon', 'information', 'number', 'system', 'datum', 'country', 'state', 'chip', 'year', 'thing'], ['right', 'fact', 'people', 'time']

Table 1: Some partitions for word clustering. The examples correspond to 17, 11, 7, and 3 partitions respectively.

We also try one experiment with the 100 most frequent words for each topic, for a total of 165 words. In this case there some words that have 0 counts for a given topic. To avoid numerical issues we add 1 to the count before normalizing the probabilities. We apply the algorithm to this data and obtain the evolution in loss of mutual information of Figure 2. A remarkable observation is that only a couple of words retains the majority of information about the topic of conversation, we are always above 93% (this is the amount if information we retain with two clusters).

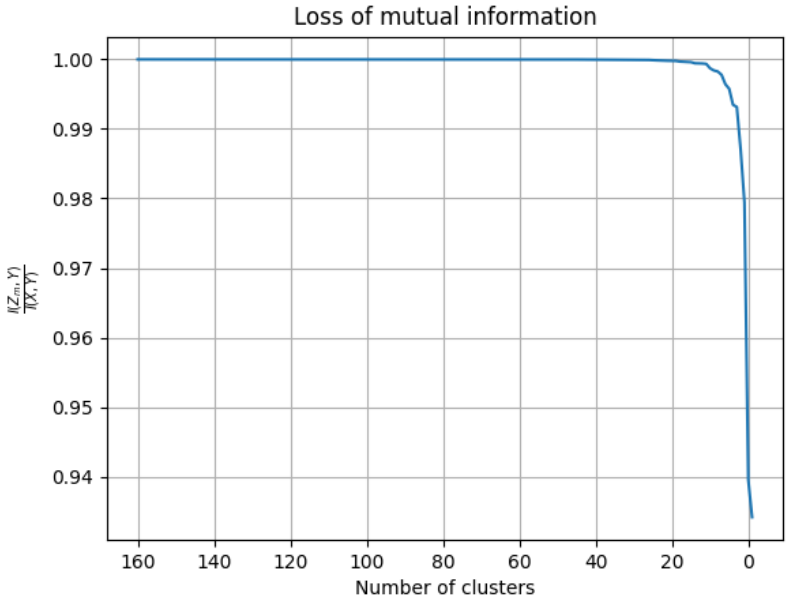

Figure 2: Evolution of the loss of information $I(Z_m, Y)$ due to the clustering

3 Conclusions

In this project we analyzed the hard-cluster limit of the Information Bottleneck method and implemented the Agglomerative Information Bottleneck algorithm, a powerful and principled method for clustering. After showcasing its usefulness in a toy example, we applied it to a real-world problem: word clustering. For that purpose, we used a dataset of online conversations about different topics. After pre-processing the raw data and applying lemmatization to each word, we were able to count the occurrence of each lemma in each type of text, thus obtaining and empirical estimation of the joint distribution of the words and the topic of the text $p(x, y)$. Once we have this distribution we can apply the algorithm and cluster the words according to their mutual information with respect to the text topic.

One advantage is that as a bypass product of the clustering procedure we get a score of how good is each partition: the loss of mutual information. This means that we can decide when to stop the clustering procedure based on how much loss of information we tolerate. This proved useful in our experiment of word clustering with 165 words. There we observed that by keeping only a few clusters we retain the great majority of the mutual information.

For the computational point of view, the main disadvantage of the Agglomerative Information Bottleneck algorithm is its dependency on the number N of possible values of the random variable X, roughly $O(N^3)$. That's because at each step (there are N steps to perform) we must compute the similarity, using the Jensen-Shannon divergence, between all the pairs $p(y|z_i)$ and $p(y|z_j)$. This similarity computation is the critical bottleneck of the algorithm when it comes to runtime efficiency.

Another limitation is that we need to have full access to the joint distribution $p(x, y)$ to be able to perform clustering. This is not always the case, however, if we can estimate it like we did for the word clustering problem, we can apply it. The estimation of the joint distribution becomes then a critical step.

References

[1] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

[2] Noam Slonim and Naftali Tishby. Agglomerative information bottleneck. In *NIPS*, 1999.

[3] Noam Slonim and Naftali Tishby. Document clustering using word clusters via the information bottleneck method. In *SIGIR '00*, 2000.

[4] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *ArXiv*, physics/0004057, 2000.