Universität Rostock

Traditio et Innovatio

Analysis and Implementation of Iterative Information Bottleneck and Agglomerative Information Bottleneck

Submitted on January 29, 2020

by

Obinna Hilary Isiwekpeni | Albert-Einstein-Straße 22 | 18059 Rostock
Matriculation Number 218204763

Supervisor:
Dipl.-Ing Clemens-Konrad Müller
Universität Rostock
Richard-Wagner-Straße 31
18119 Rostock-Warnemünde

Lead Supervisor:
Prof. Dr.-Ing. Volker Kühn
Universität Rostock
Richard-Wagner-Straße 31
18119 Rostock-Warnemünde

MASTERARBEIT
FAKULTÄT FÜR INFORMATIK UND ELEKTROTECHNIK

Universität Rostock Traditio et Innovatio

FAKULTÄT FÜR INFORMATIK UND ELEKTROTECHNIK

Institut für Nachrichtentechnik, Lehrstuhl für Nachrichtentechnik, Prof. Dr.-Ing. Volker Kühn

Specialization Module EE

Analysis and Implementation of Iterative Information Bottleneck and Agglomerative Information Bottleneck

Description:

Due to constantly increasing data rates in modern communication systems, new techniques have to be considered to process the data. Since data detection in communication engineering and classification in computer science represent the same problem, techniques from machine learning, pattern recognition, statistical signal processing or artificial intelligence could also be interesting for communication engineering. One step is to cluster the data into a defined number of groups. Since the number of groups is smaller than the cardinality of the original data, clustering can also be seen as compression. If the compression rate falls below the entropy of the data, it is not possible to reconstruct it without loss of information. With the compressed data representation (clusters), the classification / data detection can be done.

One approach for clustering originates from information theory and is called "Information Bottleneck (IB)". It tries to compress a noisy observation such that it preserves most of a predefined relevant information, i.e. the original source signal. While many of the common clustering techniques use application-specific distortion measures like mean squared error (MSE) or the Euclidean distance, the information bottleneck method is based on the mutual information.

The IB method can be implemented by different algorithms. In this thesis the iterative information bottleneck and agglomerative information bottleneck algorithm have to be implemented. It shall be investigated how the achievable relevant mutual information and rates depend on the number of clusters, how complex both algorithms are as a function of the number of clusters, and how their convergence behavior is.

Chair: Prof. Dr.-Ing. Volker Kühn
Supervisor: Dipl.-Ing. Clemens-Konrad Müller
Email: cm575@uni-rostock.de

Abstract

In recent times, there has been an increase in the relationship between Machine Learning and Data Communication. Clustering algorithms have some practical applications in Data Communication. There are times when data is to be transmitted from a sender to a receiver and the data is to be compressed where the compressed data should contain as much details as contained in the original information. Clustering is quite similar to compression, so it can be applied in Data Communication. One of such application is the Information Bottleneck Method. The Information Bottleneck seeks to pass data through a compact bottleneck representation. There are several algorithms used to implement the method but this report focuses on two of the algorithms which are Iterative and Agglomerative IB algorithms. The Iterative IB employs an iterative method that iterates over fixed point equations until a convergence criterion is reached. On the other hand, the Agglomerative IB employs a greedy clustering technique to find a clustering tree that goes in a bottom-up fashion.

Contents

List of Abbreviations

IB	Information Bottleneck
IIB	Iterative Information Bottleneck
AIB	Agglomerative Information Bottleneck
KL	Kullback-Leibler Divergence
JS	Jensen-Shannon Divergence
SNR	Signal-to-noise ratio

List of Symbols

$H(Y)$	entropy of a variable y
$\mathbf{A}$	an arbitrary matrix A
β	langrange multiplier
$I(X;Y)$	mutual information beween X and Y
C	channel capacity
$D_{KL}(p_1 \| \| p_2)$	Kullback-Leibler Divergence between two distributions
$JS_{\Pi}[p_1, p_2]$	Jensen-Shannon Divergence between two distributions

List of Figures

Chapter 1

Introduction

Communications Engineering and Computer Science have certain things in common with respect to data transfer and communication. Data detection and classification can be seen as similar problems. In Computer science, we can use certain Machine Learning algorithms to solve such classification problems. In Communications, such problems can be solved from the Information Theory perspective by a method known as Information Bottleneck (IB). There are different algorithms that are used to implement the IB, two of which will be investigated in depth in this report. They are Iterative IB and Agglomerative IB. The basic idea of the IB method is squeezing data through a bottleneck. This is a data compression process. Given a source data which can be mapped using any of the mapping schemes (Phase Shift Keying, Amplitude Shift Keying or Quadrature Amplitude Modulation). The data is passed through an Additive White Gaussian Noise (AWGN) channel. The source data has a certain cardinality which is assigned to it. The output of the AWGN channel which is mixture of the noise and source signal has to be quantized with a quantizer which will be designed. We design the quantizer such that the output of the quantizer contains as much information as possible of the source signal. In essence, we want to maximize the mutual information between the output of the quantizer and source alphabet while keeping the mutual information between the input and output of the quantizer within a certain level.

1.1 Motivation and Goal

The number of clusters or the cardinality affects the performance of the different IB algorithms. The goal is to investigate how the cardinality affects the mutual information and rates of the Iterative IB and Agglomerative IB. The complexity of each algorithm with respect to the cardinality will be investigated as well their convergence behaviour.

1.2 Report Outline

Chapter 2

Basic Theory

This chapter gives an overview of the basic ideas of the building blocks of this report. A review of probability distributions is done after which basic concepts from information theory which are relevant for the implementation are introduced.

2.1 Probability Density Function

The probability density function is an important probability function that will be useful for the implementation of this pre-thesis. We shall look at the probability density function (pdf) of the distributions used in the implementation.

2.1.1 Uniform Distribution

A process is said to be uniformly distributed if all the outcomes of the process are equally likely. For example, if we have a sample space X of size n, that is uniformly distributed where $0 < n < \infty$ then the pdf is given as

$$p(x) = \begin{cases} \frac{1}{n} & \text{for all } x \in \mathcal{X} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

2.1.2 Gaussian Distribution

The pdf of a gaussian distribution is given as

$$p(x) = \frac{1}{\sqrt{2\pi\sigma_x^2}} \mathrm{e}^{-\frac{(x-\mu_x)^2}{2\sigma_x^2}} \tag{2.2}$$

where σ_x^2 and μ_x represent the variance and mean respectively

2.2 Concepts from Information Theory

2.2.1 Entropy

Entropy is a measure of uncertainty in a given probability distribution. Assuming we have a collection of documents and a person wants to read a single document from the collection, we would like to guess the document chosen. Without any prior knowledge,

every document is equally likely. We could assume that the longer documents have a higher probability while the shorter documents have a lower probability. From the assumption, if all the documents the same number of words, then they are all equally likely. Entropy vanishes as soon as the outcome of a process is certain. The entropy of a set is maximized when all elements in the set are equally likely.
Let Y be a discrete random variable distributed according to $p(y)$. The entropy of Y is defined by

$$H(Y) = H[p(y)] = -\sum_{y} p(y) \log p(y) \tag{2.3}$$

2.2.2 Mutual Information

Given a random variable X with prior probability distribution $p(x)$ and some random variable Y with probability distribution $p(y)$, the mutual information is the information common to X and Y. Reconsidering our illustration of entropy above, assuming we also consider the distinct words in the collection of documents represented by X and the collection of documents as Y. Here, we would not only consider the prior probability distributions but also the joint distribution $p(x, y)$ which indicates the probability that a random word in the collection is equal to $x \in \mathcal{X}$ while the particular document is $y \in \mathcal{Y}$.
Mutual information is the reduction of uncertainty of X due to the knowledge of Y and also can be defined as the information gain about X due to observation of Y.

$$I(X;Y) \equiv I[p(x,y)] = \sum_{x} \sum_{y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{2.4}$$

2.2.3 Channel Capacity

Channel capacity is the maximum rate at which data can be transmitted through a channel. It can be obtained by maximizing the mutual information $I(X;Y)$ based on the statistical properties of the source signal. According to the channel coding theorem of Shannon, reliable communication can only be achieved up to the channel capacity.

$$C = \sup_{Pr\{X\}} I(X;Y) \tag{2.5}$$

2.2.4 Kullback Leibler Divergence

It is also known as the relative entropy and it is a measure of similarity between two distributions. It is not a symmetric measure as it is not commutative. Given two probability distributions $p_1(x)$ and $p_2(x)$, the Kullback Leibler (KL) divergence between them is defined as

$$D_{KL}(p_1||p_2) = \sum_{x} p_1(x) \log \frac{p_1(x)}{p_2(x)} \tag{2.6}$$

Mutual Information between two distributions can be represented in terms of their Kullback Leibler Divergence. This relationship is given as

$$I(X;Y) = D_{KL}[p(x,y)|p(x)p(y)] \tag{2.7}$$

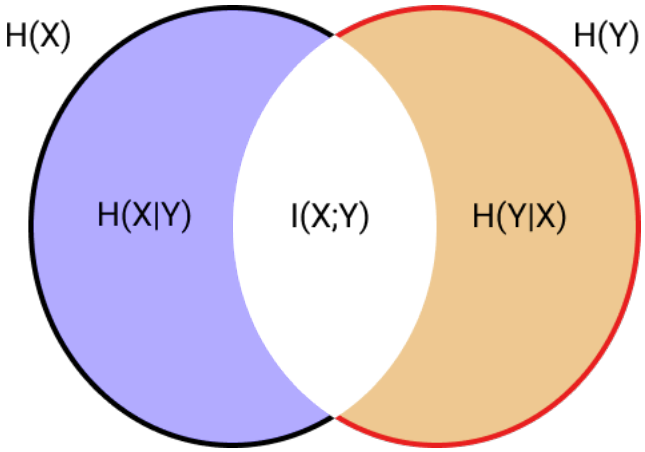

Fig. 2.1: Venn Diagram Illustrating Entropy and Mutual Information: H(X) represents the entropy of X which in this case is the source signal. H(Y) represents the entropy of Y which is the sink. H(X|Y) is the entropy of X conditioned on Y. It is also called equivocation. It is the amount of information lost during transmission. H(Y|X) is known as the irrelevance i.e information not originating from the source. I(X;Y) which is the intersection of the two circles represents the mutual information

2.2.5 Jensen Shannon Divergence

This is another measure of distance between two distributions which is important for this topic. It is symmetric as opposed to KL divergence. The Jensen Shannon (JS) divergence between two probability distributions $p_1(x)$ and $p_2(x)$ is defined as

$$JS_{\Pi}[p_1, p_2] = \pi_1 D_{KL}[p_1||\bar{p}] + \pi_2 D_{KL}[p_2||\bar{p}] \tag{2.8}$$

where $\Pi = \{\pi_1, \pi_2\}, 0 < \pi_1, \pi_2 < 1, \pi_1 + \pi_2 = 1$ and $\bar{p} = \pi_1 p_1 + \pi_2 p_2$

2.2.6 Markov Chain

Given 3 alphabets $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ with the relation $\mathcal{X} \to \mathcal{Y} \to \mathcal{Z}$, since Z depends on Y, it cannot provide any "new" information about X, except the information already given by Y.

Chapter 3

The Information Bottleneck Method

Chapter 4

Implementation

This chapter shall outline the processes and steps used in implementing the Iterative IB algorithm as well as the Agglomerative IB algorithm. Python3 was used to implement both algorithms. Numpy and Matplotlib are the relevant python libraries which was used for this implementation. The Iterative IB and Agglomerative IB both have similar initial implementations just before running their individual algorithms. To keep things simple, we would start with the general implementation common to both algorithms before moving on to the individual algorithms.

4.1 General Implementation

The implementation began with importing the relevant libraries needed. The libraries imported include numpy for numerical analysis, matplotlib for plotting and data visualization, time in order to measure the time it takes for each algorithm to run. A mapper class was also imported which was used for mapping the source signal from binary to Amplitude Shift Keying (ASK) values.
The signal-to-noise ratio (SNR) used was 6 dB. The cardinality i.e. the number of elements of the input signal X was set to 4 and real values were used for the source signal to make the investigation simple enough and not trivial by choosing a lower cardinality. The cardinality of Y which is a noisy observation of X was set to be 64 which is large enough to represent X.
The cardinality of the compressed signal Z which should be as large as the cardinality of X was set to 64 for the Iterative IB. For the Agglomerative IB, the cardinality of Z was set to be the same cardinality of Y which is 64. However the cardinality of Z was varied in both algorithms in order to investigate the effect on the performance of each algorithm.
The input signal X was modelled to be uniformly distributed that means each alphabet of the source signal is equally likely. The channel was modelled to be an Additive White Gaussian Noise (AWGN) channel. The output Y of the channel is a sum of the input signal and the noise which is Gaussian distributed with zero mean and variance(power) σ_N^2. The noise is Gaussian because it is assumed to be the sum of a large number of random processes which results in a Gaussian/Normal distribution approximately if none of the individual process has a dominant effect on the variance. This is known as the *Central Limit Theorem*. Hence the output of the signal is also gaussian distributed. The signals were modelled to follow the Markov chain $\mathcal{X} \to \mathcal{Y} \to \mathcal{Z}$ such that $\mathcal{Z}$cannot

contain more information about $\mathcal{X}$ than what is already in $\mathcal{Y}$.

Parameter	Description	Value
SNR	Signal-to-Noise Ratio	6 dB
N_x	Cardinality of X	4
N_y	Cardinality of Y	64
N_z	Cardinality of Z	32 for Iterative IB 64 for Agglomerative IB
alphabet	Input Alphabet	4-ASK
σ_X^2	Variance of source signal	1
$p(x)$	Probability of source signal	Uniform distribution
$p(y)$	Probability of output signal	Gaussian distribution
$p(z\|y)$	Quanitizer	Initialiazed as a uniform quantizer
β	Lagrangian multiplier	0.1 to 5, 5 to 400
ϵ	convergence parameter	10e $-$ 4

4.1.1 Implementation of the Probability distributions

The probability $p(x)$ of the source signal X was uniformly distributed while the probability $p(y)$ of the output signal Y was gaussian distributed. Y had to be discretized as the quantizer to be designed can only process discrete values. Then the probability of Y given X represented by $p(y|x)$ which in this case is the probability of the noise(AWGN) was computed. It was ensured that the sum of the conditional probability $p(y|x)$ across Y yielded 1. Also the conditional probability of X given Y $p(x|y)$ was obtained using Bayes rule which is given as

$$p(x|y) = p(y|x)\frac{p(x)}{p(y)} \tag{4.1}$$

The joint probability $p(x, y)$ of X and Y was computed by the multiplication of $p(x)$ and $p(y|x)$. The probability $p(y)$ of Y was obtained by summing the joint probability $p(x, y)$ across the X. The quantizer $p(z|y)$ was initialized to be a uniform quantizer and was used to calculate $p(z)$.
Attention was paid to the order in which the dimensions of the matrices were assigned in order to avoid confusions. There were some situations in which the dimensions had to be expanded in order to have similar dimensions to carry out operations. For example in the computation of $p(x|z)$, $p(z|y)$ and $p(x, y)$ needed to be multiplied. Knowing the cardinality of each of the signals, the matrices were not of matching dimensions, hence the operation could not be performed leading to the use of the *tile* and *expanddims* functions of the numpy library. The cardinality of Z, N_z was the first dimension, the cardinality of Y, N_y was the second dimension while the cardinality of X, N_x was the last dimension. Hence all matrices had the dimension (N_z, N_y, N_x).

4.2 Iterative IB

This section details the implementation specific to the Iterative IB algorithm. The quantizer $p(z|y)$ was initialized to be a uniform quantizer with *zeros* and *ones* forming a step function. The Iterative IB involves iterating through three fixed point equations and applying an update step for the next iterations. At each step, two of the distributions are kept constant and then the algorithm that minimizes the IB function:

$$L[p(z|y)] = I(Y;Z) - \beta I(X;Z) \tag{4.2}$$

At the $i+1$'th iteration, the algorithm applied an update step:

$$P^{(i+1)}(z|y) \leftarrow \frac{P^i(z)}{Z^{i+1}(y,\beta)} \mathrm{e}^{-}\beta D_{KL}(p(x|y)||P^i(x|z)) \tag{4.3}$$

where $Z^{i+1}(y,\beta)$ is a normalization factor. $P^i(z)$ and $P^i(x|z)$ were calculated using $P^{(i)}(z|y)$

$$P^i(z) = \sum_y p(y) P^{(i)}(z|y) \tag{4.4}$$

$$P^i(x|z) = \frac{1}{P^i(z)} \sum_y P^{(i)}(z|y) p(x,y) \tag{4.5}$$

The algorithm was started with a fixed β value and smaller cardinalities of Y and Z in order to ensure it runs successfully, then gradually the cardinalities were increased and also a range of beta was used. The convergence parameter was set to $10\mathrm{e}-4$ which was used to compare the Jensen-Shannon divergence. A *while* loop was used to run the algorithm until the condition in which the Jensen-Shannon divergence was less than or equal to the convergence parameter. The Jensen-Shannon divergence measured the similarity between the previous quantizer with the current quantizer and if the value of the JS divergence between the two quantizers was less than or equal to the convergence parameter, then the present quantizer was chosen and then the Relevant Information $I(X;Z)$ and Compression Information $I(Y:Z)$ was calculated. The Relevant Information $I(X;Z)$ is upper bounded by the Mutual Information between X and Y $I(X;Y)$, while the Compression Information $I(Y:Z)$ is upper bounded by the entropy of Y $H(Y)$.

4.3 Agglomerative IB

This section details the implementation specific to the Agglomerative IB algorithm. The algorithm uses a clustering technique to find a clustering tree in a *bottom-up* fashion. This algorithm aims at maximizing the function

$$L_{max} = I(X;Z) - \beta^{-1} I(Y:Z) \tag{4.6}$$

which is same as minimizing equation 4.2. The cardinalities of Y and Z at the start of the algorithm were set to 64 and the aim was to carry out clustering until the desired cardinality of Z which was set to 32 was achieved. To reduce the cardinality of Z, two

values of Z represented by z_i and z_j were iteratively merged into a single value $\bar{z}$. This was done by iterating through $p(z|y)$ which was initialized to be a uniform quantizer, hence yielding:

$$p(\bar{z}|y) = p(z_i|y) + p(z_j|y), \forall x \in \mathcal{X} \tag{4.7}$$

$\bar{z}$ is simply a union of z_i and z_j. With the Markov chain in mind, $p(\bar{z})$ was obtained by:

$$p(\bar{z}) = p(z_i) + p(z_j) \tag{4.8}$$

Also, $p(x|\bar{z})$ was obtained by:

$$p(x|\bar{z}) = \pi_i \cdot p(x|z_i) + \pi_i j \cdot p(x|z_j) \tag{4.9}$$

where

$$\Pi = \pi_i, \pi_j = \frac{p(z_i)}{p(\bar{z}}, \frac{p(z_j)}{p(\bar{z}}, \tag{4.10}$$

is the *merger distribution.*
For each pair of Z that was merged, the merger cost was calculated which is the difference between the values of L_{max} before and after the merger. This procedure was done for all the possible mergers of Z until the cardinality of 32 was reached.
The difference between the values of L_{max} for every merger was calculated using the formula:

$$\Delta L_{max}(z_i, z_j) = p(\bar{z}) \cdot \bar{d}(z_i, z_j) \tag{4.11}$$

where

$$\bar{d}(z_i, z_j) = JS_{\Pi}[p(x|z_i), p(x|z_j)] - \beta^{(} - 1) JS_{\Pi}[p(y|z_i), p(y|z_j)] \tag{4.12}$$

From equation 4.11, it can be observed that the merger cost is simply a multiplication of the merged probability $p(\bar{z})$ and the distance between them given $\bar{d}(z_i, z_j)$.