Lab Work 3

André Pedrosa [85098], Filipe Pires [85122], João Alegria [85048]

Algorithmic Information Theory
Department of Electronics, Telecommunications and Informatics
University of Aveiro

December 29, 2019

1 Introduction

This report aims to describe the work developed for the third and final assignment of the course of 'Algorithmic Information Theory', explaining all scripts developed by us and presenting the results we considered most relevant Dataset regarding the quality of the solutions.

The programs implemented in Shell have the purpose of .....Dataset Dataset Along with the description of the solutions, we also discuss ..... All code developed is publicly accessible in our GitHub repository:Dataset `https://github.com/joao-alegria/TAI` .Dataset

2 Image Classification

The challenge we are tackling in this paper is the one presented in (*1*), which is basically a image classification problem with a special approach. Instead of doing the process chosen by most people in charge of solving similar problems, consisting in using some from of feature extraction or some kind of feature learning commonly achieved through some type of machine learning algorithm, the launched challenge aims to reach a similar goal taking only in consideration the notion of algorithmic information.

Algorithmic information, also known as algorithmic entropy or Kolmogorov complexity, is the measure of information in a given data portion based on the theoretical algorithm that would be necessary to create that same data portion. Being a quite abstract definition, an example can clarify the concept: taking in consideration the string A=”0101010101” and string B=”0010100110”, it is much simpler to define string A than string B, since for A we can simple say that it is the string ”01” repeated 5 times, where for string B it’s harder and we probably need to say the entire string to describe it. This means that in reality string A as less information than string B. To achieve the algorithmic information of each one, the thought process is the same but now the string descriptions are made with algorithms, where the string can be defined be the following algorithms:

```
def stringA():
  for i in range(5):
    print("01", end="")
```

```
def stringB():
  for letter in "0010100110":
    print(letter, end="")
```

Which by a quick iteration analysis its obvious that `stringA()` is less complex than `stringB()`, needing only 5 iterations compared with 10 iterations, respectively, confirming that string B has more information than string A. This can be directly correlated with data compression, since compressors take advantage of the data repetition as dispensable data, and thereby compress the information by removing repetition but storing an indicator of that repetition.

With this concepts defined it’s more clear how the image classification will be done. Using the Normalized Information Distance(NID) defined by:

$$NID(x,y) = \frac{max\{K(x|y), K(y|x)\}}{max\{K(x), K(y)\}} \tag{1}$$

Where *K(x)* defines the Kolmogorov complexity of the string x and *K(x—y)* defines the Kolmogorov complexity of the string x given string y. Unfortunately, because the Kolmogorov complexity is non-computable this formula is only theoretical since there is no algorithm that can compute any string complexity every time. For that reason compressors are used to achieve

approximations of the formula result. The studied approximations are Normalized Compression Distance(NCD) and Normalized Conditional Compression Distance(NCCD) that will be further detailed in the following subsections.

2.1 NCD Approximation

This approximation, which follows the Formula 2 is frequently used to replace the NID. By using known general purpose compressors, such as gzip, lzma, tar, ..., the Kolmogorov complexity can be estimated by the number of bits the resulting compressed file has.

$$NCD(x,y) = \frac{C(x,y) - min\{C(x), C(y)\}}{max\{C(x), C(y)\}} \tag{2}$$

In Formula 2, *C(x)* represents the number of bits the compression of x contains and *C(x,y)* represents the number of bits the compression of x and y concatenated. This concatenation can be performed by several ways, the most common is to simply perform an union on both, but can be as complex as interlacing the two data sources. Following the formula, results close to 0 indicate similarity and close to 1 dissimilarity. The only restriction to this approach is that the compressor used must follow the condition $C(x,x) \simeq C(x)$, without that the results of NCD may not be reliable nor true.

2.2 NCCD Approximation

This approach is yet another approximation to the NID formula, this time using conditional compressors to achieve their estimation. Conditional compressors are normally implemented with finite-context models and take in consideration the previous occurring terms to compress the new term. This is a more direct approximation of the NID and, in general, generate better results than the NCD. Defined by:

$$NCCD(x,y) = \frac{max\{C(x|y), C(y|x)\}}{max\{C(x), C(y)\}} \tag{3}$$

Where *C(x)* defines once again the number of bits the compression of the object x by a given conditional compressor and $C(x|y)$ represents the number of bits the compression of x needs given that y occurred previously ($C(y|x)$ has the same logic). Due to the consitional aspect of this approach, *C(x)* can be a special case of $C(x|y)$ were y is equal to the empty string, causing that $C(x) = C(x|\epsilon)$.

3 Dataset

Lorem ipsum ...

4 Experiment

Lorem ipsum ...

4.1 Tested Compressors

Zip is a compression and file packaging utility for many operating systems, like Unix, VMS, MSDOS, etc.. It is analogous to a combination of the Unix commands tar(1) and compress(1) and is compatible with PKZIP (Phil Katz's ZIP for MSDOS systems). It performs a lossless compression that can achieve 3:1/2:1 ratios in text files. The standard compression method is `deflate`, which uses a combination of LZSS and Huffman coding.

Gzip is a single-file/stream lossless data compression utility. This compression algorithm uses Lempel-Ziv coding (LZ77) to achieve its reduction factors. Gzip will only attempt to compress regular files. In particular, it will ignore symbolic links. Typically, text such as source code or English is reduced by 60-70%. Compression is generally much better than that achieved by LZW, Huffman coding, or adaptive Huffman coding.

The Lempel–Ziv–Markov chain algorithm, aslo known as LZMA, is an algorithm used to perform lossless data compression. Developed by Igor Pavlov, was first used in the 7z format of the 7-Zip archiver. This algorithm uses a dictionary compression scheme somewhat similar to the LZ77 algorithm published by Abraham Lempel and Jacob Ziv in 1977 and features a high compression ratio (generally higher than bzip2).

Bzip2 is a free and open-source file compression program that uses the Burrows–Wheeler algorithm. It only compresses single files and is not a file archiver. This algorithm compresses most files more effectively than the older LZW and Deflate compression algorithms, but is considerably slower. LZMA is generally more space-efficient than bzip2 at the expense of even slower compression speed, while having much faster decompression.

ZPAQ is an open source command line archiver for Windows and Linux. It uses an append-only format which can be rolled back to an earlier state to retrieve older versions. It supports fast incremental update by adding only files whose last-modified date has changed since the previous update. It compresses using deduplication and several algorithms (LZ77, BWT, and context mixing) depending on the data type and the selected compression level.

Ppmd, or Prediction by partial matching is an adaptive statistical data compression technique based on context modeling and prediction. PPM models use a set of previous symbols in the uncompressed symbol stream to predict the next symbol in the stream. PPM algorithms can also be used to cluster data into predicted groupings in cluster analysis.

5 Results & Discussion

Lorem ipsum ...

6 Conclusions

After completing the assignment, we drew a few conclusions regarding our solutions and .....

Lorem ipsum ...

In terms of code organization and readability, we made sure our repository was as well structured as possible and our code properly commented and documented. The base folder contains a *README* file for basic instructions. All code is in the *src* folder.

References

1. Armando J. Pinho, *AIT: Lab Work no.3*, University of Aveiro, 2019/20.