Neural Data Compression

Paul Bagourd

Chair of Statistical Field Theory
Supervisor: Vassilis Papadopoulos
Professor in charge: Clément Hongler

Introduction

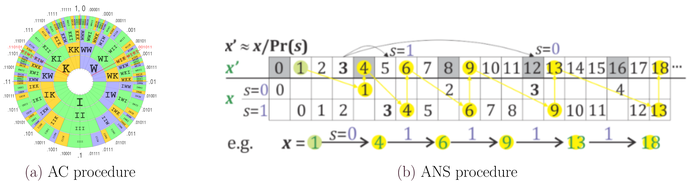
As the exchange of information has increased exponentially with the computer revolution, it has become essential to simplify and store information suitably to spread it seamlessly. In this context, Neural Data Compression methods are any methods that use artificial neural networks to model and compress data smoothly. This thesis presents two specific methods by considering theoretical and practical perspectives.

Source coding

Source of information

The data source to compress is described by a stochastic process $\{X_t\}_{t>0}$ with outcomes belonging to a source alphabet $\mathcal{X}$. We can then concatenate the outcomes x_t into sequences $S = x_1 \dots x_n$. Source coding is the task of mapping sequences from a source of information to sequences belonging to another alphabet, usually the binary alphabet. In the context of data compression, source coding is equivalent to answering the question:

How to map sequences of symbols into short binary sequences or small natural numbers?



(a) AC procedure (b) ANS procedure

Arithmetic Coding

Let us say the source alphabet is made of the letter symbols "W", "I", "K", appearing with respective probabilities $1/4, 1/2, 1/4$. Arithmetic Coding (AC) encodes any sequence S by updating a starting interval $I_0 = [0, 1]$ whose updated length is proportional to the probability of the encoded symbol. Using the final interval I_n, we can send a binary sequence of length $\log_2 l(I_n)$ whose float value lies in I_n. Such a binary sequence is guaranteed to exist due to the size of the final interval. With the encoded binary sequence and the probabilities of the symbols, the decoder first searches in I_0 for the subinterval I_1 containing the corresponding encoded float value. The symbol associated with this subinterval is the first decoded symbol of the sequence. Then, the decoder repeats this step to find the following encoded symbols in the same order as the encoder proceeded. For this reason, we say that AC is queue-like.

Asymmetric Numeral Systems

The second method corresponds to the Asymmetric Numerical System (ANS) and, more specifically, a generalisation of the positional numeral system that we use to read symbol digits, for instance, "365" :

$$0 \to_{\times 10+3} 3 \to_{\times 10+6} 36 \to_{\times 10+5} 365 \to 101101101_2$$
$$365\%10^1 \to \text{"5"}, 365//10^1 = 36, 36\%10^1 \to \text{"6"}, 36//10^1 = 3, 3\%10^1 \to \text{"3"}$$

ANS is a skewed version in which we store sequence information by increasing a natural number N (sequence $S =$ "01111" into number x in the ANS figure to the right). The increase is inversely proportional to the probability of the symbol to be encoded. As for the usual symmetric case where the symbol probability is constant (10^{-1} in the decimal system), explicit encoding and decoding formulas for varying symbol probabilities exist for ANS. Moreover, since the last encoded symbols are the first decoded, ANS is stack-like.

Adaptation to the source

Since the encoded length should be as small as possible on average, what truly happened with AC and ANS is conceding a significant growth for rare symbols to allow for a beneficial, minor increase when encoding frequent symbols. This trade-off minimises the weighted average size according to the symbol probabilities, thus reducing the compression ratio since sequences with frequent symbols end up with a shorter encoded length. We say that we code a symbol x_k according to a probability distribution p.

Entropy codings

Both AC and ANS coding are entropy codings, being optimal in the sense that given a set of symbol frequencies, or equivalently, a specific uncertainty in the information source, it is guaranteed that the source cannot be better compressed by any other coding scheme. In particular, in the information theory realm, this means that the growth of the encoded sequence length $\log_2 l(I_k)$ for AC or $\log_2 N_k$ for ANS when encoding symbol x_k corresponds to the information content $-\log_2 p(x_k)$:

$$L_k - L_{k-1} \approx -\log_2 p(x_k)$$

Source modelling

We explained why getting closer to the proper frequencies of a source is essential to suitably minimising the average size of a compressed sequence. However, symbols are usually not independent from each other, and their actual frequency depends on the neighbour symbols in the sequence. Hence, modelling the source of information to learn about symbol frequencies in different contexts is crucial.

Artificial Neural Networks

Artificial Neural Networks learn any features from data by processing information from one layer of neurons to the next, similar to how the brain processes electric signals using neurons. The learning works through trial and error, updating the weights of the synapses linking the neurons after each trial to minimise the value of a cost function designed for a specific task.



Predictive models

A predictive model learns the context of symbols from a source by trying to predict the frequency of the following symbol given the past symbols in a sequence $(x_t)_{t>0}$.

$$p(x_k | x_{k-1}, \dots, x_{k-c})$$

The transformer architecture performs well in this regard, and we use it to implement AC.

Latent variable models

Sometimes, predictive models are not suited to model data, and we can use latent variables belonging to a latent space Z. Latent variables do not need to have particular significance, but they can be helpful to model complicated distribution by breaking it up into simple parts with the following specification if $\mathcal{Z}$ is continuous

$$p(x_k) = \int_{z \in \mathcal{Z}} p(x_k|z) p(z) dz$$

$\mathcal{Z}$ can be understood as an alternative dataset carrying some structure of the original dataset. Unfortunately, we do not have access to $p(x_k)$ because the integral is costly to compute, but the model provides the probabilities $\{p(z), q(z|x_k), p(x_k|z)\}$ where q is an approximation of $p(z|x)$. Bayes's theorem gives

$$\tilde{p}_z(x_k) \coloneqq \frac{p(z) q(x_k|z)}{p(z|x_k)} \approx p(x_k)$$

Learning a probability distribution

If the objective is to learn the parameters $\boldsymbol{\theta}$ of a probability distribution, then the cost function is designed to maximise the likelihood of the training set X

$$\mathcal{C}_X(\theta) = -\frac{1}{|X|} \sum_{x \in X} \log p(\boldsymbol{\theta}|x)$$

Bits-back coding and chaining

Predictive models and AC match well since the frequency needed when encoding the k-th symbol can be directly accessed with $p(x_k|x_{k-1}, \dots, x_{k-c})$. However, when using a latent variable model, the method to leverage the underlying structure learned by the model is more subtle. Clearly, we should exchange information between $\mathcal{X}$ and $\mathcal{Z}$. Let us say we wish to encode a data point x_0. We could encode first some z_0 according to probability $p(z_0)$ and then encode x_0 according to $p(x_0|z_0)$. If one uses AC and the probabilities $p(z), p(x|z)$ given by the latent variable model, the receiver can first recover z_0 and then x_0. However, the remaining data z_0 is useless for him, and still, it is some information that incurred an undesired growth of the encoded length L compared to having directly encoded x_0 using $p(x_0)$, although not accessible.
In fact, there is a way to mimic a direct use of $p(x_0)$ if the encoder could somehow use the extra information z_0 stored in the current encoded sequence to generate a next latent z_1 for a future data point x_1. This can be done by decoding from the current encoded sequence according to the distribution $q(z_1|x_1)$. This generation would subsequently decrease L as follows

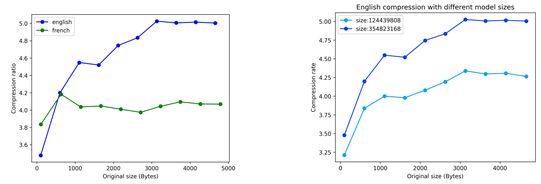
$$0 \to_{p(x_0|z_0)} L_1 \to_{p(z_0)} L_2 \to_{q(z_1|x_1)} L_3 \to_{p(x_1|z_1)} L_4 \to_{p(z_1)} L_5$$

where black/orange arrows are encoding/decoding steps. Consequently, $L_1 < L_2, L_3 < L_4 < L_5$ and $L_3 < L_2$ because we extract some information from the encoded sequence to generate z_1. More precisely $L_2 - L_3 \approx -\log q(z_1|x_1)$, the information content of z_1 knowing x_1. This forward process is reversible by simply exchanging encoding and decoding steps, with one encoding step mid-way: the "bits-back". However, since we first need z_1 to decode x_1 according to $p(x_1|z_1)$, we must go in the backward direction and use a stack-like coding scheme such as ANS.

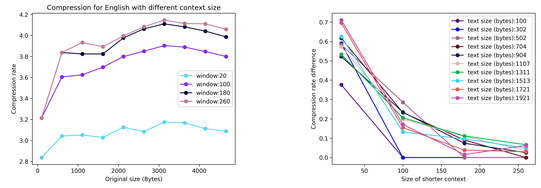
Implementation results

Text experiments

We chose the piece 'A Midsummer Night's Dream' by William Shakespeare and a biography of Ernest Dupuy titled 'Victor Hugo, son oeuvre poétique'. For the trained model, we used OpenAI models gpt2 (124 million parameters) and gpt2-medium (354 million parameters) that can be found on the Hugging Face platform. Since those models are predictive, we used the encoding scheme AC.



From what we see on the left graph, the compression performs better in English than in French. This is because gpt2 and gpt2-medium were mainly trained on English web pages. Hence, the model better knows what is more likely to come next if it is an English text, having a better knowledge of this language. The right graph shows how gpt2-medium with more neurons better compresses English text.



The graph to the left shows the influence on the past context (or window) of size c that the model uses to make predictions. The compression ratio increases as the size increases linearly. Intuitively, this reflects that words in a neighbourhood are firmly connected, so the probability of saying the next word can be very accurate, given we know the past words. However, the right graph also shows the importance of past words fades as we move away in the text. On the ordinate axis are the differences in the compression ratio between a context experiment of size s (shorter context) and a context experiment of size $s + 80$ for s, going from 20 up to 260 with step 80.

Image experiments

We first experimented on four 32×32 images RGB with 512 colours per pixel using the predictive model ImageGPT-small from OpenAI, a GPT-2-like model with 76 million parameters trained on the ImageNet dataset to predict the next pixel value in raster order.

	Image 1	Image 2	Image 3	Image 4
Compression ratio	14.1	13.9	13.1	18.4
Runtime (seconds)	155	160	164	166

Image 4 has a better compression rate because it is actually the image of a cat, which is common in the ImageNet dataset. The other images are mountain landscapes from Switzerland, so their compression is approximately the same, structurally being the same.
We then tested our implementation of bits-back coding on the MNIST dataset consisting of written digits with 256 shades of greys and 32×32 pixels. We trained a continuous latent variable model called a variational autoencoder (VAE) on the same dataset for this experiment.

	"0"	"1"	"2"	"3"	"4"	"5"	"6"	"7"	"8"	"9"
Compression ratio	3.94	7.91	4.04	4.58	4.93	3.71	4.20	5.49	4.90	4.29

We see that the digit "1" is more compressible, probably because of its more straightforward structure.

Conclusion

This thesis shows how neural methods and entropy codings can significantly compress texts and images. The compression depends on various factors, such as the specification of the models, the training set, and the dataset to be compressed.

References

[1] A. Said. Introduction to arithmetic coding–theory and practice. arXiv:2302.00819, 2023.

[2] J. Townsend, T. Bird, and D. Barber. Practical lossless compression with latent variables using bits back coding. arXiv:1901.04866, 2019.