VARIABLE RESOLUTION MARKOV MODELLING OF SIGNAL DATA FOR IMAGE COMPRESSION

Mark Trumbo

IBM T.J. Watson Research Center
P.O. Box 218, 04-250
Yorktown Heights, NY, 10598
`trumbo@watson.ibm.com`

Jacques Vaisey

School of Engineering Science
Simon Fraser University
Burnaby, BC, V5A 1S6
`vaisey@ensc.sfu.ca`

ABSTRACT

Traditionally, Markov models have not been successfully used for compression of signal data other than binary image data. Due to the fact that exact substring matches in non-binary signal data are rare, using full resolution conditioning information generally tends to make Markov models learn slowly, yielding poor compression. However, as is shown in this paper, such models can be successfully applied to non-binary signal data compression by continually adjusting the resolution and order to minimize the codelength of the past samples in the hope that this choice will best compress the future samples as well, a technique inspired by Rissanen's Minimum Description Length (MDL) principle. Performance of this method meets or exceeds current approaches.

1. INTRODUCTION

It is commonly felt [1] that Markov models are inappropriate for compression for all but binary signal data. This opinion is based on the fact we are looking for exact matches in the data when we do Markov modelling. However, measurements of physical processes (such as images) are generally noisy representations of processes that have infinite resolution. For example, consider the case where binary data is corrupted by white Gaussian noise and then digitized to 256 levels of grey. Although there are only two levels in the generating mechanism, the effect of the noise will be to create many additional contexts. Each histogram in each context will thus see correspondingly fewer samples and will take longer to generate accurate probability estimates.

Existing variable order techniques designed for text compression [2] counteract the learning problem for Markov models to some extent, but still do not give us sufficient control over the number of states in the model. We will show that we need to vary both the order and the resolution of the conditioning information as we compress signal data.

This paper begins by determining the optimal resolution/order of a set of test sources consisting of a synthetic sources as well as six common 8-bit grayscale images converted to a one-dimensional sequences via raster scans: `pentagon` (512×512), `lena` (256×256), `photographer` (256×256), `barb` (512×512), `mandrill` (512×512), and a (512 × 512) `texture` pattern. We then present some novel methods for choosing the optimal resolution/order combination without an exhaustive search or even a pre-scan of the data.

As a final introductory note, the algorithms described in this paper are adaptive. Adaptive Markov models, most often used in conjunction with arithmetic coding [3], start with no information about the source, accept data sequentially, are presented with each symbol in the sample only once, and modify the way they compress in response to the history. Adaptive algorithms are attractive in that it is unnecessary to perform a pre-scan of the data, and no side information need be sent.

2. PRELIMINARIES

Before proceeding, we need to introduce the terms *state weight*, which is simply the number of contexts in a Markov modeller and *permutation*, which is an ordered set of integers [3], $p = \{p_1, p_2, \cdots, p_n\}$, where n is the model order. The source to be coded is given by $x^t = x_1, x_2, \cdots, x_t$, where x_i is the i^{th} symbol, and we define the order n context of x at symbol t using permutation p as the sequence $x_{t-p_1}, x_{t-p_2}, \cdots, x_{t-p_n}$. For the context to be causal, each element of p must be greater than zero.

Now, let us define the *resolution modification* as an ordered set of integers in $[0, r]$, $m = \{m_1, m_2, \cdots, m_n\}$, where r is the source resolution in bits. This modification is applied to the previously defined context resulting in the resolution reduced context

$$(x_{t-p_1}2^{m_1-r}, x_{t-p_2}2^{m_2-r}, \cdots, x_{t-p_n}2^{m_n-r})\,. \qquad (1)$$

This work was supported by IBM and the Natural Sciences and Engineering Research Council of Canada.

Note that in performing the resolution reduction, we truncate the fractional part of the individual products. An order n modeller will be denoted by

$$(r_1 = r - m_1, r_2 = r - m_2, \cdots, r_n = r - m_n)\ , \qquad (2)$$

that is, by the resolution in bits of the conditioning information. Choosing a specific value of n and a specific sequence m results in $N = 2^{\sum_{i=1}^{n}(r-m_i)}$ possible contexts for the current symbol. The state weight is thus N.

3. FIXED ORDER/FIXED RESOLUTION MODELLING

Consider adaptively modelling a sample generated by an unknown stationary source that generates symbols according to some distribution. The appearance of each symbol in the sequence is then used to increment the appropriate histogram count for the current context. These counts are then used to produces estimates of the probability distributions, which are then passed on to an arithmetic coder. In this work, we use the non-linear estimation technique described by Williams [1].

Since they get “hits” more often, we expect that the lower order joint probabilities will be more accurate than the higher order ones; however, exactly what n should be used to code the input after t samples depends on the distribution. Solving this problem is what motivated the development of the currently existing variable order techniques. For instance, the “Universal Markov Coding” (UMC) algorithm [2] tends to use low order modellers at the beginning of a sample and work up to higher order modellers as more samples are seen. This works better than just using a high order modeller because they yield uniform predictions (which are bad unless the underlying distributions are uniform) until enough symbols have been processed.

Based on the success of the other simplifying methods like splitting the input into bit planes [4] or running variable order models, we expect that varying the resolution of the conditioning information in a Markov model will improve the performance. To test this conjecture, 2^{16} samples were generated using an AR(2) model driven by white Gaussian noise with $\sigma_N^2 = 64$. The samples were then quantized to $r = 8$ bits of resolution and translated to fall in the range $[0, 255]$. The parameters for the AR(2) model, $\phi = \{0.01, 0.89\}$, were chosen to demonstrate some properties of the modellers described later. An exhaustive set of order 2 modellers spanning all resolutions for the order 1 and order 2 conditioning information were then run on the source.

The procedure used will now be described. For each sample x_t and for each model, the per-symbol codelength (the negative logarithm of the probability assigned to the symbol by the model) was saved. Then, after the entire sample x had been coded by each model, the performance of all models was compared based on the sum of the per-symbol codelengths for each model. The modeller with the lowest overall codelength was called the *optimal static* choice. If more than one modeller performed equally well, the one with the lower state weight was chosen. The performance of the modellers is shown in Figure 1.

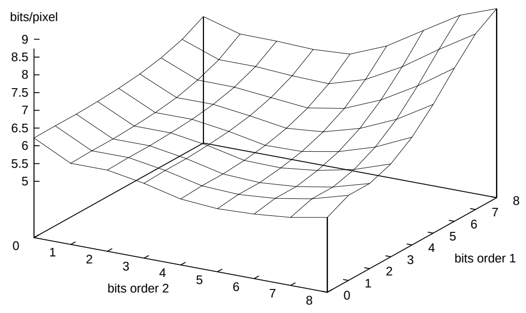

Figure 1: FOFR Performance on the AR(2) Source

The best performance attained by a single modeller on this source was with the $(0, 5)$ modeller at 5.19 bits/pixel. That is, the best order 2 modeller for this sample ignored the order 1 conditioning information altogether, and used 5 bits for the order 2 conditioning information. Referring to the correlation sequence of this test source, this choice makes sense in that the sample immediately prior to the current sample is not strongly correlated to the current sample. This optimal static choice represents a savings of 21.6% over the $(0, 0)$ model, 6.4% over the $(0, 8)$ model, 25.6% over the $(8, 0)$ model, and 40.6% over the $(8, 8)$ model. These models were chosen as comparison because they represent models that one might naively choose if one did not know about the effect of using reduced-resolution conditioning information.

The optimal static order 2 resolution modification was determined for each image in the image test set. The average maximum and minimum improvements over the test set when compared to the $(0, 0)$, $(0, 8)$, and $(8, 0)$ models were 24% and 6.2% respectively.

4. VARIABLE RESOLUTION MODELLING

Based on the above results, a system could determine the optimal resolution via a pre-scan through the data, and transmit n and m as side-information. But is this fixed resolution modeller the best we can do? Additionally, due to the fact that the data may be non-stationary and the optimal resolution choice may not be constant, is it possible to avoid the pre-scan?

Our experiments have shown that lower resolution modellers typically perform best in the very early portion of the sample; however, as more data is seen, the higher resolution modellers overtake them. Hence, we would like to initially use low state weight modellers and then move to higher state weight modellers as more data is seen, choosing the resolution in such a way as to optimally exploit the correlation in the sample. One way of doing this is to run many modellers in competition and to choose between them based on their recent performance metric as in Williams' [1] multimodal data compression (MMDC) algorithm. Specifically, each model maintains a recent performance measure

$$H_{i,t} = -\sum_{j=1}^{t} \rho^{t-j} \log p_i(x_j) \tag{3}$$

where $p_i(x_t)$ is the probability assigned to sample x_t by model i and ρ corresponds to the half-life of the performance information. The half-life and the parameter ρ are related via the equivalent expressions

$$\rho = e^{\log \frac{1}{2}/h} \leftrightarrow h = \frac{\log \frac{1}{2}}{\log \rho} \,. \tag{4}$$

The half-life means that the effect of the performance of model i at time $t-h$, on the total performance measure at time t, is one half the effect of the performance of model i at time t.

In the VR approach, the family of models is simply one that contains all possible resolutions for a given value of n (which automatically selects different orders from zero to n, since resolutions can be zero). However, one problem with this approach is that some subset of these modellers may never be chosen at all, and yet (a lot of) resources will be consumed updating them. We thus adopt a different approach that grows models as needed.

We begin modelling our signal source using only the order 0 model. Since it is our conjecture that the state weight of the optimal model choice increases monotonically with the number of samples seen, a good guess for the next optimal modeller is the one that has an incrementally higher state weight than the current modeller. However, having chosen the arbitrary maximum order n, there are in fact n such modellers, namely $(1, 0, \cdots, 0), (0, 1, \cdots, 0), \cdots, (0, 0, \cdots, 1)$, each of which has a state weight of 2. To be thorough, the algorithm should create these n higher state weight modellers and run the order 0 modeller in competition with them to determine when to change to a modeller with a higher state weight. Assuming that there exists some redundancy in the signal to be exploited, one of the n modellers, will be chosen, most likely the one that uses the conditioning information at the offset with the highest correlation with the current symbol being coded. Once the algorithm selects this modeller, it should again create and run in competition those modellers that have an incrementally higher state weight compared to the selected modeller.

From that starting point let us define the VR algorithm as follows:

1. read next sample and code it with the *best* modeller (initially the order 0 one)
2. update the performance metric for all existing modellers
3. generate list of modellers performing the best by comparing the performance metrics of all existing modellers
4. choose the one modeller from list with lowest state weight and designate it the *best* modeller
5. for all modellers on the list, grow "child" modellers with incrementally higher state weight
6. if there are more samples go to step 1 else quit

A modeller is allowed to grow if it is found to be the one that performs the best at the current instant t. If there are several that are performing identically, all are allowed to grow.

Model growth is defined as creation of the n modellers with incrementally higher state weight than the parent model. Thus the n children of modeler $(3, 2, 5)$ would be $(4, 2, 5)$, $(3, 3, 5)$, and $(3, 2, 6)$. Of course, the maximum resolution of the modellers is limited to the source resolution. According to this definition, a node could have been created from several different parents. All of these newly created modellers have the same state weight, but could be anything from order 1 to order n (in the conventional understanding of the term) because the components with resolution zero are skipped.

The least frequently used model is destroyed if the maximum number of models is exceeded while a growth is occurring, or when the algorithm has exceeded the allowed memory consumption. This has implications on the coding of non-stationary data, since the model growth is always in the direction of increasing state-weight. A possible modification to the VR algorithm would be to note the derivative of the per-symbol codelength and to use this information to allow the state-weight to shrink. This is a topic for further research.

Note that the model growth rules described above are in fact implementing a form of permutation function definition, since those members of the function that do not help as much as others will have fewer bits of resolution allocated to them by the algorithm.

The algorithms presented above differ in a significant way from PPMI [5]. Only the models that are performing well "grow" and only to the extent in resolution and order

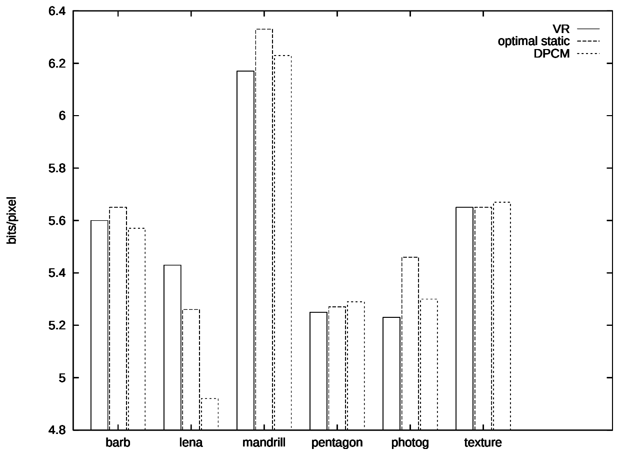

Figure 2: Performance on the Test-Set

that is necessary. Instead of using an arbitrary criterion for how many samples is enough to constitute a reliable set for a given order and resolution, the algorithm uses the recent codelength to decide between models of varying resolution, a technique that is an application of the minimum description length (MDL) principle.

4.1. Experiments

The performance of the VR modeller on the image test-set is shown in Figure 2. The the half life h was set to 128 samples, the maximum number of models to 128, the maximum memory usage to 16 MB, and the maximum order to 2. The permutation used in these experiments considered the pixel immediately to the left of the current pixel first, and then the pixel immediately above, as is appropriate and common in image coding. For comparison, the optimal static and DPCM performance is also given. The DPCM results are based on order 3 predictor coefficients determined by a pre-scan of the data – the codelength of the residual, as determined by an asymptotically adaptive order 0 model, is given as the rate.

Despite our initial hesitation about the performance of the algorithm on non-stationary sources, the VR algorithm performed well overall on the test set, resulting in codelengths below the statically chosen optimal value for all sources except `lena`, for which the codelength was approximately 3.2% higher. The performance is also slightly better than that obtained using DPCM. Improved performance over DPCM is expected to be possible if the image data is kept in its 2D form rather than being converted into a raster.

5. CONCLUSIONS

We have investigated some methods for solving the Markov model learning problem. Instead of adjusting the number of states in the model using just the model order, we have the ability to vary the resolution of the conditioning information as well. In fact, to obtain the performance obtainable from methods like DPCM, we *must* vary the resolution this way. Due to their flexibility, we see that on signal sources that cannot be well represented by a linear model, fixed order/variable resolution models can outperform DPCM. However, it appears that many natural image sources can be well represented by linear models. The VR techniques implement a useful form of permutation selection, effectively ignoring data that does not help code the source.

6. REFERENCES

[1] R. Williams, *Adaptive Data Compression*. Kluwer, 1991.

[2] J. Rissanen, "Complexity of strings in the class of Markov sources," *IEEE Transactions on Information Theory*, vol. 32, pp. 526–532, July 1986.

[3] G. Langdon and J. Rissanen, "Compression of black-white images with arithmetic coding," *IEEE Transactions on Communications*, vol. 29, pp. 858–867, 1981.

[4] Joint Bi-level Image Experts Group, *CCITT Draft Recommendation T.82/ISO Draft International Standard 11544*, April 1992.

[5] P. Howard and J. Vitter, "New methods for lossless image compression using arithmetic coding," in *Proceedings, Data Compression Conference, Snowbird, Utah* (J. Storer and J. Reif, eds.), pp. 257–266, 1991.