Machine Learning for Video Compression: Macroblock Mode Decision

Christoph H. Lampert
German Research Center for Artificial Intelligence (DFKI)
67663 Kaiserslautern, Germany
christoph.lampert@dfki.de

Abstract

Video Compression currently is dominated by engineering and fine-tuned heuristic methods. In this paper, we propose to instead apply the well-developed machinery of machine learning in order to support the optimization of existing video encoders and the creation of new ones. Exemplarily, we show how by machine learning we can improve one encoding step that is crucial for the performance of all current video standards: macroblock mode decision.

By formulating the problem in a Bayesian setup, we show that macroblock mode decision can be reduced to a classification problem with a cost function for misclassification that is sample dependent. We demonstrate how to apply different machine learning techniques to obtain suitable classifiers and we show in detailed experiments that all of these perform better than the state-of-the-art heuristic method.

1. Introduction

1.1. Video encoding

Digital video is usually available in compressed form. Neither DVDs nor digital television would be possible without video compression, because in an uncompressed format, digital video exceeds the capacities of currently affordable hardware: a typical movie in DVD quality would require 30 MB/s of constant throughput, for a total of 160 GB of storage. Current video compression standards like MPEG-2, MPEG-4, and H.264 only specify the syntax of the compressed bitstreams, but not the process of how to generate such streams. This allows for different implementations of video encoders while ensuring compatibility of the resulting output streams.

Studying the encoding process in more detail, it becomes apparent that it is necessary to make several thousand decisions per second in order to achieve a high compression ratio at a decent encoding speed. Surprisingly, none of the currently available video encoders use machine learning techniques to do so. Instead, they rely on heuristic criteria with parameters that have been fine-tuned carefully over many years (see e.g. [2, 7]).

In this paper, we propose a change of paradigm. We leave the typical engineering approach of selecting a model based on trial-and-error experiments and afterwards adjusting the parameters manually until good performance is achieved. Instead, we show that as good and even better results can be achieved much faster, by using machine learning techniques to guide and assist the software development process.

A video encoder is a large and complex piece of software. Therefore, we restrict ourselves for the moment to only one particular aspect of video encoding, which to us seems prototypical for whole set of problems and which is common to all state-of-the-art video compression methods: macroblock mode decision.

1.2. Macroblock mode decision

The current compression standards evolved from common ancestors, and they share many underlying principles. For example, to encode an image of a video sequence it is split into a regular pattern of quadratic blocks, typically 16x16 pixels in size. For each of these so-called *macroblocks*, different mechanisms to code the image information can be applied. One option is to encode the block in a JPEG-like manner; this is referred to as *INTRA*-mode. Alternatively, a predictor block from a position in the previous image can be chosen and the pixel-wise difference between the current block and the prediction is encoded. This is called *INTER*-mode.[1]

Depending on the actual encoding standard, many other modes might be possible, for example splitting the block into smaller subblocks or using more than one block for prediction. In the theoretical treatment this plays no particular role. In our practical experiments, we restrict ourselves to

[1]This is of course an oversimplified description of the INTRA- and INTER-modes. As a technically correct reference [3] should be used.

INTER and INTRA modes, because this simplifies the description. Also, this makes the results less dependent on the actual standard chosen, since INTER and INTRA are the only modes that all current standards share.

The problem of *mode decision (MD)* is to decide which coding mode to use for each block in a sequence. This typically is done in an online fashion, sequentially from left to right and from top to bottom. Although simple to state, MD plays a crucial role in the video encoding task. A bad method for mode decision can easily reduce the coding efficiency by 50% or more.

2. Mathematical Formulation

2.1. MD as an optimization problem

To solve the mode decision problem, we need a quality measure that allows us to judge which encoding mode is best in a given situation. Such a measure has to reflect the two aspects that are influenced by the decision process: the amount of image distortion caused and the number of bits needed for signaling the mode and for representing the compressed image information. Typically, both values are combined using the Lagrangian formalism into a *rate-distortion*-measure [5]:

$$\mathrm{rdbits}(x;m) = \mathrm{bits}(x;m) + \lambda \cdot \mathrm{dist}(x;m), \qquad (1)$$

where bits is the number of bits required to encode a block x in mode m and dist measures the L^2-distortion caused in the image by doing so. λ is a Lagrangian multiplier that depends on the encoding standard as well as on several other factors, but in our context we can assume it to be constant.

The mode decision problem now emerges as a minimization problem: for a current block x we aim to find the encoding mode m such that $\mathrm{rdbits}(x;m)$ is minimal, i.e. we are looking for the mapping $x \mapsto m(x)$ given by

$$m(x) := \operatorname*{arg\,min}_{m \in \{m_1,\dots,m_c\}} \mathrm{rdbits}(x;m). \qquad (2)$$

The rdbits measure can be computed, therefore the problem can be solved by an exhaustive search. Some encoders indeed have an "optimal quality" mode that performs mode decision in exactly this way: it encodes the block in every permitted mode, counts for each case the resulting bits and measures the distortion. Finally, it chooses the mode with lowest resulting rdbits-score. However, this procedure is slow and speed is a crucial feature in modern video encoders. Therefore it is logical to search for other solutions.

2.2. MD as a classification problem

Since Equation 2 means a selection from a finite number of classes, we can also state MD as a classification problem, for example in the Bayesian framework. To do so, we treat the block x as a random variable from a configuration space $\mathcal{X}$, and the optimal encoding mode y as a discrete random variable from the set of all possible modes $\mathcal{M} = \{m_1, \dots, m_c\}$. The objective is to find a classification function $\alpha : \mathcal{X} \to \mathcal{M}$ that minimizes the probability of error, which is

$$E(\alpha) = \int_{\mathcal{X}} P(\alpha(x) \neq y | x) p(x) dx. \qquad (3)$$

where $p(x)$ is the probability density for a block x to occur and $P(\alpha(x) \neq y|x)$ is the probability that for a given sample x the classifier α decides for the wrong class. If we are given a number of sample blocks $(x_i)_{i=1,\dots,N}$ with known classes $(y_i)_{i=1,\dots,N}$, this expression translates into the simpler empirical classification error rate

$$ER(\alpha) = \frac{1}{N} \sum_{i=1}^{N} \delta[\alpha(x_i) \neq y_i] \qquad (4)$$

where $\delta[\,]$ denotes the indicator function that is 1 if the enclosed condition is true and 0 otherwise. We can now use classical machine learning techniques to find a suitable classifier α, see Section 3.

2.3. Sample specific cost functions

The classifier from the previous section tries to make as few errors as possible. This corresponds to a risk minimizing approach where for each block a $0/1$ cost function is used. However, in video encoding blocks differ strongly in how important their correct treatment is for the resulting output quality and the encoding efficiency. For one block, the difference between INTER and INTRA mode can be negligible, whereas for other blocks one encoding mode is clearly superior to the other. These difference is reflected by the corresponding rdbits-values having similar values or differing by several thousand units.

In the Bayesian framework the correct way to incorporate this block diversity is to minimize the *expected cost* instead of the expected error rate:

$$C(\alpha) = \int_{\mathcal{X}} c(\alpha(x)|x) p(x)\, dx \qquad (5)$$

where each block has its specific cost function $c(m|x) := \mathrm{rdbits}(x;m)$ for each mode m. With a set of samples as before, this gives rise to the empirical cost functional:

$$EC(\alpha) = \frac{1}{N} \sum_{i=1}^{N} c(\alpha(x_i)|x_i), \qquad (6)$$

which again must be minimized by finding a good classifier α.

Classification problems where the cost for misclassification depends not only on the class information but also on the current sample are uncommon. Typical machine learning techniques do not include support to train a classifier with respect to this criterion.

However, for the two-class problem, in which we are most interested, we can simplify the problem: at first, we notice that the problem of minimizing $EC(\alpha)$ is unchanged when the constant term $EC(\alpha_{opt})$ is subtracted, where α_{opt} is the optimal classifier as it follows from Equation 2. The new term is

$$EC(\alpha)-EC(\alpha_{opt})=\frac{1}{N}\sum_{i=1}^{N}c(\alpha(x_i)|x_i) - c(\alpha_{opt}(x_i)|x_i)$$

which we can rewrite to

$$= \frac{1}{N}\sum_{i=1}^{N} w_i \; \delta[\alpha(x_i) \neq y_i] \tag{7}$$

with $w_i = |c(m_1|x_i) - c(m_2|x_i)|$.

After a suitable normalization, w_i can be interpreted either as a probability for the sample x_i to occur, or as a multiplicity how often the sample x_i appears in the sample set. Both interpretations allow to adapt existing machine learning techniques to our situation, see Section 3.

3. Applying Machine Learning

In the previous section, we saw different formulations of the mode decision problem. We now show how different machine learning techniques can be used to address them. Our main concern (apart from a high accuracy) is fast classification, because in a video encoder, MD is performed online and often under real-time conditions. The time for training is less important, since we can perform the training offline and only include the resulting classifier with fixed parameters into the encoding software.

3.1. Linear classifiers

Linear classifiers that work directly in the feature space, for example the classical perceptron, are very fast to evaluate. Basically, each decision just requires one scalar product of the features with a weight vector. Although data in MD is unlikely to be linearly separable, we include a linear classifier in our test. Another reason for this is that the heuristic mode decision methods that are used by current encoders typically are linear as well.

We have used a simple perceptron classifier for the learning with minimal error rate (Equation 4). For risk-minimization (Equation 7), we modified the training step to use importance sampling during the training phase, where the probability for a sample to be chosen is proportional to its weight w_i.

3.2. Decision trees

Decision trees are very fast classifiers as well. Additionally, they have the advantage that they can directly be transformed into source code in any programming language, essentially just forming a cascade of *if*-statements.

We have used the OC1-software [4] to train a classifier that minimizes the error rate, using its axis-parallel mode and the Gini-Index [1] as impurity measure. Additionally, we modified the OC1-source code to include the weights w_i as multiplicities with which the samples occur, allowing to also train for cost-minimization.

3.3. Neural networks and other classifiers

Artificial neural networks are known to be universal, meaning that correctly set up, they can learn any decision function to arbitrary high precision. However, their speed for classification usually is lower than for the previous two methods, because the typically trigonometric activation functions require more complex calculations in each node. We included a simple feed-forward network with backpropagation training and 8 hidden nodes into our tests. As in the linear case, we used importance sampling to optimize for cost instead of for the error rate.

Many other machine learning techniques exist and might even perform superior in terms of classification accuracy. Typical candidates are support vector machines (SVMs) and nearest neighbor classifiers (NNCs). However, these are even slower in evaluation: for SVMs, the classification time depends on the number of support vectors which tends to grow large when the number of training samples is high. For NNCs, classification requires access to all training data available, which is completely out of question in our setup.

4. Experimental Results

To test if machine learning techniques can improve the current mode decision routine in video encoders, we need some reference to compare our results with. For this, we chose the XviD MPEG-4 encoder[2] because of two reasons: Firstly, it is a Free Software/Open Source project, therefore allowing to modify the source code and also to re-distribute the results of our modifications. Secondly, XviD is generally accepted as one of the best MPEG-4 codecs available and has been so for several years. That way, we are

[2]http://www.xvid.org/

positioned against a "strong opponent" and if we find improvements, we know that the encoding process itself was enhanced and not just some inferior implementation.

We extended XviD's mode decision routines to create logfiles during the encoding step. They contain the 4 features that are currently used in XviD's MD process and 4 additional features that are also simple to extract and that we believe can improve the decision process. This results in one 8-dimensional feature vector per 16×16 block[3]. Using XviD's mode of operation in which it performs optimal mode decision at the expense of reduced speed, we also generated ground truth class data and the number of *rdbits* necessary to encode in each of the possible modes.

We then tested the performance of our learning based mode decision algorithm on a total of 18 standard test sequences in CIF resolution (352×288 pixels)[4]. Most of them show rather static video scenes, like for video conferencing, but also some difficult scenes like from a football match are included. The experiments were performed in a leave-one-out manner: for each of the 18 sequences, we created a test set that consisted of all blocks of the sequence and a training set that consisted of the blocks of all remaining sequences. We then trained and tested the perceptron, the decision tree, and the neural network, each once with the error rate (ER) and once with the expected cost (EC) as optimization criterion. The leave-one-out method ensures that for each sequence the classifier is tested only on data that was not used during training. This is meant to simulate the situation in a real video encoder, where once the algorithm is implemented, it has to work for all possible input.

The results of our experiments are presented in Table 1. Rows correspond to the sequences and columns to the classifiers tested. For comparison, the results of the rate-distortion optimal classifier are given as well. As indicator of classification quality, we give the relative *rdbits*-increase or decrease in percent, with the XviD heuristic decision serving as reference. Thus, negative values are better and any value below zero means a better performance than XviD's.

That almost all numbers in the table are negative shows that using machine learning techniques, one can indeed improve the mode decision quality compared to the heuristic method. Three of the sequences (*akiyo, container* and *waterfall*) are exceptions here, because XviD already behaves optimal or almost so. For all other, already the simplest classifier, the linear perceptron, improves the classification accuracy. By optimizing for the minimal expected cost instead of the error rate, the number of *rdbits* is reduced some more.

Decision trees and artificial neural networks as more complex classifiers improve the result further. Optimizing for expected cost seems to have only a minor influence here. Both methods reach almost the same performance: the gain is about half of the theoretical optimum. An exception is the *husky*-sequence, where none of the classifiers manages to excel. We believe that this is due to the source material distribution and the leave-one-out training: *husky* contains much irregular motion, which is not well reflected in the rest of the training material.

Because of its higher classification speed, the decision tree appears preferable to the neural network in practice. But it will require more testing in a realistic encoding environment to decide if the increase in encoding efficiency is sufficient to justify the additional computational effort compared to the linear decision at all.

5. Summary

We have shown that machine learning techniques can improve the mode decision process of a state-of-the-art video encoder. All classifiers tested as alternatives to the XviD heuristic had a positive influence on the coding efficiency, even the simple perceptron, especially when it is combined with relevance sampling to optimize for a sample dependent cost function. The classification quality for the decision tree and the neural network are higher, but they also come at a higher computational cost.

To keep our analysis simple and applicable for all encoding formats, we have demonstrated the effect only for the most basic case of INTRA vs. INTER decision. The most recent video standards have many more possible modes; for example, the HDTV-standard H.264/AVC has as many as 33 different way to encode a block even in its simplest profile [6]. An optimal decision by exhaustive search is clearly out of question here, and an efficient multi-class classifier is required instead.

It is our belief that with upcoming HDTV-techniques, video compression will gain even more importance within the next years and machine learning should try to claim this field for itself. This means not only mode decision as we have demonstrated here, but also other parts of the encoding process, like rate control or motion estimation. In this paper, we have described a machine learning system that assists the programmer during the software development process. It finds a static classifier solution through data-driven training that can then be implemented into the encoder. But this is only a first step. It is our belief that in future, adaptive techniques will emerge, where the learning algorithms are built into the system and training is performed online, making the encoder adaptive to its current input material. We will examine these questions in future work.

[3]To the reader who is familiar with the technicalities of video encoding: XviD uses as features the encoding modes of top and left neighboring block, the block's variance and the SAD for the best motion vector found. We added the class of the top right neighbor and the Euclidean lengths of best motion vector, the prediction vector and of their difference vector

[4]available e.g. from http://media.xiph.org/video/derf/

	Perceptron		Decision tree		Neural network		rdbits-
sequence	ER	EC	ER	EC	ER	EC	optimal
akiyo	0.00	-0.01	-0.01	0.00	-0.00	-0.00	-0.01
bus	-0.40	-0.63	-0.59	-0.69	-0.65	-0.70	-1.17
coastguard	-0.01	0.03	-0.03	-0.02	-0.02	-0.01	-0.11
container	0.00	0.00	0.00	0.00	0.00	0.00	0.00
deadline	-0.04	-0.04	-0.07	-0.06	-0.06	-0.05	-0.10
flower	-0.02	-0.03	-0.12	-0.15	-0.08	-0.13	-0.24
football	-1.75	-3.33	-3.22	-3.56	-3.24	-3.65	-4.88
foreman	-0.03	-0.12	-0.14	-0.15	-0.12	-0.13	-0.32
hall	-0.03	0.01	-0.21	-0.22	-0.16	-0.19	-0.42
husky	-0.49	-0.11	-0.43	-0.44	-0.47	-0.38	-6.51
mobile	-0.01	-0.01	-0.02	-0.02	-0.01	-0.02	-0.06
mother	-0.01	-0.04	-0.07	-0.07	-0.07	-0.05	-0.11
news	0.01	-0.35	-0.37	-0.38	-0.28	-0.28	-0.67
paris	-0.03	-0.34	-0.37	-0.37	-0.33	-0.35	-0.51
silent	-0.11	-0.13	-0.32	-0.31	-0.29	-0.29	-0.50
stefan	-0.52	-0.53	-0.72	-0.73	-0.66	-0.67	-1.01
tempete	-0.14	0.07	-0.26	-0.24	-0.21	-0.21	-0.34
waterfall	0.00	0.00	0.00	0.00	0.00	0.00	0.00
average	*-0.19*	*-0.29*	*-0.39*	*-0.41*	*-0.37*	*-0.40*	*-0.93*

Table 1. Percentual rdbits-increase or decrease of the mode decision classifiers, relative to the performance of the XviD heuristic. Each classifier was trained using the error rate (ER) and the expected cost (EC) as measure.

References

[1] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[2] T. D. H. Du. Macroblock mode decision for H.264. In *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, pages 167–172, 2005.

[3] MPEG2. Information Technology - Generic Coding of moving pictures and associated audio information. ISO/IEC 13818-2, International Standards Organization, Geneva, Switzerland, 2000.

[4] S. K. Murthy, S. Kasif, and S. Salzberg. A System for Induction of Oblique Decision Trees. *J. Artif. Intell. Res. (JAIR)*, 2:1–32, 1994.

[5] G. J. Sullivan and T. Wiegand. Rate-Distortion Optimization for Video Compression. *IEEE Signal Processing Magazine*, 15(6):74–90, November 1998.

[6] G. J. Sullivan and T. Wiegand. Video Compression - From Concepts to the H.264/AVC Standard. In *Proc. the IEEE, Special Issue on Advances in Video Coding and Delivery*, volume 93(1), pages 18–31, 2005.

[7] D. Wu, F. Pan, K. P. Lim, S. Wu, Z. G. Li, X. Lin, S. Rahardja, and C. C. Ko. Fast intermode decision in H.264/AVC video coding. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 15(7), pages 953–958, July 2005.