Edge Adaptive Prediction for Lossless Image Coding

Wee Sun Lee
Department of Computer Science
School of Computing
National University of Singapore
Singapore 119260
Republic of Singapore

Abstract

We design an edge adaptive predictor for lossless image coding. The predictor adaptively weights four directional predictor together with an adaptive linear predictor based on information from neighbouring pixels. Although conceptually simple, the performance of the resulting coder is comparable to state of the art image coders when a simple context based coder is used to encode the prediction errors.

1 Introduction

State of the art lossless image coders typically perform prediction followed by entropy coding of the prediction errors. The prediction stage is usually used to reduce the complexity of the probability modelling done by the entropy coder, hence reducing the modelling cost of coding [7].

Images which we wish to compress are usually nonstationary and can be reasonably modelled as smooth and textured areas separated by edges. In this paper, we consider a simple edge adaptive prediction scheme which is designed to perform well around simple edges and smooth areas of images. The predictor achieves good performance by adaptively weighting four directional predictors and an adaptive linear predictor based on information obtained from the neighbours of the pixel being predicted. The four directional predictors we use are simply the previous pixels in four directions while the adaptive linear predictor is a linear function of the neighbouring pixels adaptively updated using the Widrow-Hoff algorithm. For the weighting scheme, we assume that the prediction errors have a Laplacian distribution and perform a Bayesian weighting based on the prediction errors in a small window of neighbouring pixels. This weighting scheme is similar to that used in [6] for predicting missing pixels in images and video. Our simulation results show that the weighting scheme is much more effective than selecting the maximum a posteriori predictor based on the prediction errors of the same neighbouring pixels. Our results also show that weighting the adaptive linear predictor with the directional predictors results in a better predictor than using either the adaptive linear predictor or the directional predictors alone.

The prediction error of our predictor is entropy coded using a simple context based entropy coder. The context is calculated by uniformly quantizing the sum of absolute errors of a small number of neighbouring pixels. With this simple context coder, the performance of our coder is comparable to the state of the art coder CALIC [8].

In contrast to the predictor in CALIC which is heuristically designed, the weighting scheme and the component predictors in our prediction scheme are conceptually quite simple. Our results also show that a simple scheme for adapting the predictor to edges is already competitive with the context tree based methods used in [7].

2 The Predictor

2.1 Weighting Scheme

Let t be the time index generated by a raster scan of an image. Let y_t denote the current pixel and let $\boldsymbol{y}_{t-1}$ denote the sequence of pixels which been scanned prior to y_t. We assume that the current pixel is generated according to the following process:

$$y_t = \hat{y}_{kt} + \epsilon$$

for some $k \in \{1, \ldots, m\}$, where k is the index of a finite set of predictors and ϵ has a Laplacian distribution $p(\epsilon) = \frac{1}{2\beta} e^{-|\epsilon|/\beta}$.

We will predict the pixel Y_t using the conditional expectation $E(Y_t|\boldsymbol{y}_{t-1})$. We can write

$$E(Y_t|\boldsymbol{y}_{t-1}) = \int Y_t p(Y_t|\boldsymbol{y}_{t-1}) dY.$$

Assumming that there is a hidden random variable taking a finite number m of states, where the predictor $\hat{y}_{kt}$ is active in state u_k, we have

$$p(Y_t|\boldsymbol{y}_{t-1}) = \sum_{k=1}^{m} p(Y_t, u_k|\boldsymbol{y}_{t-1}) = \sum_{k=1}^{m} p(Y_t|u_k, \boldsymbol{y}_{t-1}) p(u_k|\boldsymbol{y}_{t-1}).$$

For each k, we perform a change of variable $\epsilon = Y_t - h_{k|\boldsymbol{y}_{t-1}}$, where $h_{k|\boldsymbol{y}_{t-1}}$ is the conditional expectation of the process when it is in state u_k. This allows us to write

$$E(Y_t|\boldsymbol{y}_{t-1}) = \sum_{k=1}^{m} \int (h_{k|\boldsymbol{y}_{t-1}} + \epsilon) p(u_k|\boldsymbol{y}_{t-1}) p(h_{k|\boldsymbol{y}_{t-1}} + \epsilon|\boldsymbol{y}_{t-1}) d\epsilon = \sum_{k=1}^{m} h_{k|\boldsymbol{y}_{t-1}} p(u_k|\boldsymbol{y}_{t-1}).$$

This suggests that we take a weighted average of all the predictions weighted by the a posteriori probability of the states. We need to calculate $p(u_k|\boldsymbol{y}_{t-1})$ for each k. We do this using a window of pixels which are neighbours to the pixel being predicted. The effect of using a window of neighbouring pixels is to allow the predictor to change according to the local statistics. Let $\sigma(i)$ denote a permutation of the natural numbers. Let $\boldsymbol{y}_{t,n}$ be a finite set of neigbours of y_t, selected by using the appropriate permutation σ and letting

$\boldsymbol{y}_{t,n} = (y_{t-\sigma(1)}, \dots, y_{t-\sigma(n)})'$. Let $\bar{\boldsymbol{y}}_{t,n} = \boldsymbol{y}_{t-1} - \boldsymbol{y}_{t,n}$ be the set of pixels in $\boldsymbol{y}_{t-1}$ which are not in $\boldsymbol{y}_{t,n}$. For this paper we take $\boldsymbol{y}_{t,n}$ to be the pixels shown in Figure 1. From Bayes rule,

$$p(u_k|\boldsymbol{y}_{t,n}, \bar{\boldsymbol{y}}_{t,n}) = \frac{p(\boldsymbol{y}_{t,n}|u_k, \bar{\boldsymbol{y}}_{t,n})p(u_k|\bar{\boldsymbol{y}}_{t,n})}{\sum_{i=1}^{K} p(\boldsymbol{y}_{t,n}|u_i, \bar{\boldsymbol{y}}_{t,n})p(u_i|\bar{\boldsymbol{y}}_{t,n})}.$$

To simplify matters, we assume that all states are equally probable given $\bar{\boldsymbol{y}}_{t,n}$, i.e. $p(u_i|\bar{\boldsymbol{y}}_{t,n}) = p(u_j|\bar{\boldsymbol{y}}_{t,n})$ for $i, j \in 1, \dots, m$. Then

$$p(u_k|\boldsymbol{y}_{t,n}, \bar{\boldsymbol{y}}_{t,n}) = \frac{p(\boldsymbol{y}_{t,n}|u_k, \bar{\boldsymbol{y}}_{t,n})}{\sum_{i=1}^{K} p(\boldsymbol{y}_{t,n}|u_k \bar{\boldsymbol{y}}_{t,n})}.$$

Calculating the a posteriori probability of the states is simple with the models we are using:

$$p(\boldsymbol{y}_{t,n}|u_k, \bar{\boldsymbol{y}}_{t,n}) = \frac{1}{2^n \beta^n} e^{-\frac{1}{\beta}\sum_{i=1}^{n} |y_{t-\sigma(i)} - \hat{y}_{k(t-\sigma(i))}|}.$$

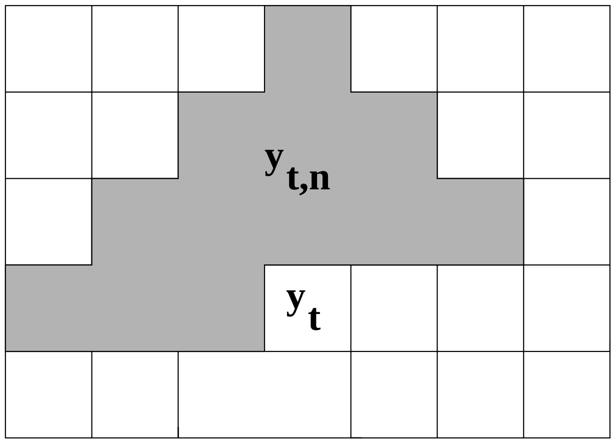

Figure 1: Current pixel and neighbouring pixels used in prediction and context modelling.

2.2 Component predictors

For the directional predictors, we use the optimal predictors for the following four processes: $y(i,j) = y(i,j-1) + \epsilon$, $y(i,j) = y(i-1,j) + \epsilon$, $y(i,j) = y(i-1,j-1) + \epsilon$, $y(i,j) = y(i-1,j+1) + \epsilon$, where i is the vertical index of the image starting from the top left hand corner of the image and going downwards, j is the horizontal index of the image and as before ϵ has a Laplacian distribution $p(\epsilon) = \frac{1}{2\beta}e^{-|\epsilon|/\beta}$. Since ϵ is zero mean, the optimal predictors are simply the previous pixels in the appropriate direction. The directional predictors are useful around edges in images.

For the linear predictor, we use the optimal predictor for the following process:

$$y_t = w_1 y_{t-\sigma(1)} + \dots + w_n y_{t-\sigma(n)} + \epsilon$$

where $y_{t-\sigma(1)}, \ldots , y_{t-\sigma(n)}$ are shown in Figure 1 and ϵ is as before. Since ϵ has zero mean, the optimal predictor is simply the linear function. The linear function should perform well in smooth areas of images.

In order to learn a suitable weight vector $\boldsymbol{w}$, we use a simple online gradient descent algorithm also known as the Widrow-Hoff algorithm. For the Widrow-Hoff algorithm, the weight vector is updated according to the following:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + 2\eta(\hat{y}_t - y_t)\boldsymbol{y}_{t,n}.$$

The learning rate η needs to be chosen small enough so that the algorithm does not become unstable. We use a value $\eta = 1/(4 \parallel \boldsymbol{y}_{t,n} \parallel^2_{max})$ where $\parallel \boldsymbol{y}_{t,n} \parallel^2_{max}$ is an upper bound for the squared norm of $\boldsymbol{y}_{t,n}$ (we use $128^2 n$ as the upper bound since the pixels range from 0 to 255 and we center it by taking away 128 before performing the predictions). This value of η is guaranteed to perform close to the best fixed value of $\boldsymbol{w}$ in a worst case sense [1, 5].

2.2.1 Boundary Conditions

In this paper, we simply pad the pixels outside the image boundaries with zeros.

3 Entropy Coder

We use a simple context based entropy coder. First, we calculate the sum of absolute value of prediction error s for a window of neighbouring pixels. We use the same neighbouring pixels $\boldsymbol{y}_{t,n}$ as before (as shown in Figure 1). Next we quantize s with a fixed quantization parameter q to a fixed number N of quantized value. Uniform quantization is used except for values of s larger than Nq which is quantized to the value $N - 1$. The quantized values of s is used as an index for selecting the entropy coder to be used for coding the prediction error of the current pixel. A similar entropy coder is used in [2]. In our simulations, we use $q = 10$ and $N = 10$ and a histogram based adaptive arithmetic coders (provided in [3]) as the entropy coders.

The aim of the context based coder is to segment the (prediction error) image in such a way that pixels with similar statistics (expected absolute errors) are coded using the same entropy coder.

4 Simulation Results

4.1 Selecting the values of β

In our models, we have one parameter β of the Laplacian distributions which has to be determined. We do this by empirically trying out several values of β over a set of images. The results for several values of β are shown in Table 1. The best values of $\beta = 20$ will be used in the rest of the paper. The performance of the coder does not seem to be very sensitive to small change of β around $\beta = 20$.

Image	$\beta = 5$	$\beta = 20$	$\beta = 50$
Balloon	2.80	2.79	2.83
Barb1	4.24	4.20	4.26
Barb2	4.52	4.47	4.50
Board	3.53	3.50	3.56
Boats	3.77	3.76	3.82
Girl	3.70	3.70	3.80
Gold	4.40	4.35	4.40
Hotel	4.29	4.24	4.26
Zelda	3.71	3.68	3.72
Average	3.88	3.85	3.91

Table 1: Bits per pixel for different values of β

4.2 Weighted Predictor Versus Maximum a Posteriori Predictor

Instead of weighting the predictors, it is possible to select the the predictor which has the maximum a posteriori probability for performing the prediction. As can be seen in Table 2, the weighted predictor outperforms the maximum a posteriori predictor.

Image	Weighted Predictor	Max a Posteriori Predictor
Balloon	2.79	2.98
Barb1	4.20	4.34
Barb2	4.47	4.62
Board	3.50	3.69
Boats	3.76	3.90
Girl	3.70	3.82
Gold	4.35	4.50
Hotel	4.24	4.41
Zelda	3.68	3.85
Average	3.85	4.01

Table 2: Bits per pixel for weighted versus maximum a posteriori predictor

4.3 Linear versus Directional Predictors

A natural question to ask is whether the adaptive linear predictor or the directional predictors are helpful for compression. Table 3 shows that weighting the predictors together improves the performance of using either just the linear predictor or just the directional predictors.

Image	Weighted Predictor	Linear Predictor	Directional Predictors
Balloon	2.79	2.97	2.83
Barb1	4.20	4.55	4.49
Barb2	4.47	4.68	4.61
Board	3.50	3.76	3.56
Boats	3.76	3.98	3.89
Girl	3.70	3.89	3.82
Gold	4.35	4.49	4.43
Hotel	4.24	4.55	4.30
Zelda	3.68	3.80	3.75
Average	3.85	4.07	3.96

Table 3: Bits per pixel for weighted versus linear and directional predictors

4.4 Comparison Against the State of the Art

We also compared our coder against the coder CALIC [8] which was the best performing coder on the test images used for evaluating proposals for the lossless JPEG standard. The performance of the coder is slightly better than the performance of CALIC for the images we tested. The results are shown in Table 4.

Image	Weighted Predictor	CALIC
Balloon	2.79	2.82
Barb1	4.20	4.42
Barb2	4.47	4.53
Board	3.50	3.56
Boats	3.76	3.84
Girl	3.70	3.77
Gold	4.35	4.39
Hotel	4.24	4.24
Zelda	3.68	3.74
Average	3.85	3.92

Table 4: Bits per pixel for weighted predictor versus CALIC.

5 Discussion

In this section, we discuss some possibilities for further work.

The Bayesian weighting scheme is known to have good worst case performance (see e.g. [4]) for online learning. We have used windowing in order to allow the algorithm to track the best predictor. In [4], different schemes with guaranteed worst case bounds are proposed for tracking the best predictor which changes with time. It may be possible to extend these schemes for tracking the best predictor in two dimension (images) for comparison with the windowing scheme.

We have followed the common practice of separating out the prediction and probability modelling stages in this work. Our prediction stage actually uses a well defined probability model for forming the prediction. It is possible to use the probability model directly for compression. However, it may be necessary to track both β and the best predictor in order to obtain good performance for image compression.

6 Conclusions

We have shown that simple edge adaptive prediction using a weighting technique is sufficient to achieve state of the art lossless image compression performance when used with a simple context based entropy coder.

7 Acknowledgements

The arithmetic coder used in this paper was written by Geoff Davis and John Danskin as part of the Baseline Wavelet Transform Coder Construction Kit [3]. The author would like to thank Phil Long for helpful discussions. Most of this work was done while the author was at the Australian Defence Force Academy.

References

[1] N. Cesa-Bianchi, P. M. Long, and M. K. Warmuth. Worst-case quadratic loss bounds for prediction using linear functions and gradient descent. *IEEE Transactions on Neural Networks*, 7(3):604–619, 1996.

[2] C. Chrysafis and A. Ortega. Efficient context-based lossy wavelet image coding. In *Proc. of Data Compression Conference*, 1997.

[3] G. Davis and John Danskin. Baseline wavelet transform coder construction kit. http://www.cs.dartmouth.deu/ gdavis/wavelet/wavelet.html.

[4] M. Herbster and M. Warmuth. Tracking the best expert. *Machine Learning*, 32(2), August 1998.

[5] J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. Technical Report UCSC-CRL-94-16, UCSC, 1994.

[6] W. S. Lee, M. R. Frater, M. R. Pickering, and J. F. Arnold. Spatial temporal concealment of lost blocks in coded video. In *International Conference on Image Processing*, 1998.

[7] M. J. Weinberger, J. J. Rissanen, and R. B. Arps. Applications of universal context modeling to lossless compression of gray-scale images. *IEEE Trans. on Image Processing*, 5(4):575–586, April 1996.

[8] X. Wu, N. Memon, and K. Sayood.
A context-based, adaptive, lossless/nearly-lossless coding scheme for continuous-tone images. http://www.csd.uwo.ca/faculty/wu/.