Predictive coding

Samples close to each other in a signal are often strongly correlated. To get an efficient coding it's always a good idea to utilize the correlation (memory).

A general *predictive coder* utilizes the signal samples up to N steps back in time to make a prediction (guess) of what the signal sample at the present time should be and then codes the difference between the prediction and the real value.

We will concentrate on *linear predictors*, where the prediction is a linear combination of old sample values. This corresponds to having an AR model of the source.

Linear prediction

Idea: We guess (predict) the signal value at time n as a linear combination of the previous N sample values.

$$\begin{aligned} p_n &= a_1 x_{n-1} + a_2 x_{n-2} + \ldots + a_N x_{n-N} = \\ &= \sum_{i=1}^{N} a_i x_{n-i} \end{aligned}$$

The difference between the real value and the predicted value, the *prediction error*, $d_n = x_n - p_n$ is quantized, possibly source coded and then sent to the receiver. The receiver reconstructs d_n, calculates p_n and can then recreate x_n.

Unfortunately, this will not work in practice!

The problem is that the receiver can only recreate a distorted version $\hat{d}_n$ of the prediction error and therefore only a distorted version $\hat{x}_n$ of the signal.

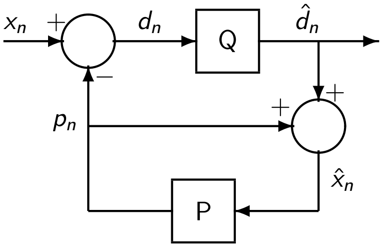
Linear prediction

In order for the predictive coder to work, the coder must perform the same prediction that the decoder can perform.
The prediction must be done from the reconstructed signal $\hat{x}_n$ instead of from the original signal.

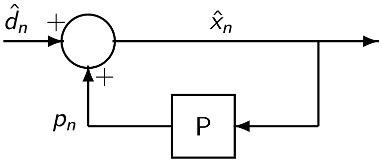
$$\begin{aligned} p_n &= a_1\hat{x}_{n-1} + a_2\hat{x}_{n-2} + \ldots + a_N\hat{x}_{n-N} = \\ &= \sum_{i=1}^{N} a_i\hat{x}_{n-i} \end{aligned}$$

The prediction error d_n is quantized and transmitted. Both coder and decoder recreate $\hat{d}_n$ and $\hat{x}_n = p_n + \hat{d}_n$.

Predictive coder and decoder



Predictive coder



Predictive decoder

Optimizing the predictor

How should we choose the predictor coefficients a_i?
Given a rate R we want to minimize the distortion

$$D = E\{(x_n - \hat{x}_n)^2\} = E\{((p_n + d_n) - (p_n + \hat{d}_n))^2\} = E\{(d_n - \hat{d}_n)^2\}$$

The quantization makes it hard to calculate optimal a_i exactly. If we assume *fine quantization*, ie that the number of quantization levels is large, we can do the approximation

$$\hat{x}_n \approx x_n$$

ie we will do our calculations as if we predicted from the original signal.
Using fine quantization we also have the approximation

$$D \approx c \cdot \sigma_d^2 \cdot 2^{-2R}$$

where σ_d^2 is the variance of the prediction error and c depends on what type of quantization we're doing and the distribution of d_n. We can thus minimize the distortion by minimizing the variance of the prediction error.

Optimizing the predictor

Variance of the prediction error:

$$\begin{aligned}
\sigma_d^2 &= E\{d_n^2\} = E\{(x_n - p_n)^2\} = \\
&= E\{(x_n - \sum_{i=1}^{N} a_i \hat{x}_{n-i})^2\} \approx \\
&\approx E\{(x_n - \sum_{i=1}^{N} a_i x_{n-i})^2\}
\end{aligned}$$

Differentiate with respect to each a_j and set equal to 0, which gives us N equations

$$\frac{\partial}{\partial a_j}\sigma_d^2 = -2 \cdot E\{(x_n - \sum_{i=1}^{N} a_i x_{n-i}) \cdot x_{n-j}\} = 0$$

Matrix description

This can be written in the form of a matrix equation

$$\mathbf{RA} = \mathbf{P}$$

where

$$\mathbf{R} = \begin{bmatrix} R_{XX}(0) & R_{XX}(1) & \cdots & R_{XX}(N-1) \\ R_{XX}(1) & R_{XX}(0) & \cdots & R_{XX}(N-2) \\ \vdots & \vdots & \ddots & \cdots \\ R_{XX}(N-1) & R_{XX}(N-2) & \cdots & R_{XX}(0) \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} R_{XX}(1) \\ R_{XX}(2) \\ \vdots \\ R_{XX}(N) \end{bmatrix}$$

and where $R_{XX}(k) = E\{x_n \cdot x_{n+k}\}$ is the auto correlation function of x_n.

Matrix description

The solution can be found as

$$\mathbf{A} = \mathbf{R}^{-1}\mathbf{P}$$

For the optimal predictor **A** we get the prediction error variance

$$\sigma_d^2 = R_{XX}(0) - \mathbf{A}^{\mathbf{T}}\mathbf{P}$$

NOTE: This formula can not be used for other choices of prediction coefficients.

Prediction gain

With fine quantization the distortion and signal to noise ratio are given approximately as

$$D_p \approx c \cdot \sigma_d^2 \cdot 2^{-2R}, \quad \mathrm{SNR}_p = 10 \cdot \log_{10} \frac{\sigma_x^2}{D_p}$$

where σ_x^2 is the variance of the original signal.
If we had quantized the original signal directly we would have gotten

$$D_o \approx c \cdot \sigma_x^2 \cdot 2^{-2R}, \quad \mathrm{SNR}_o = 10 \cdot \log_{10} \frac{\sigma_x^2}{D_o}$$

The difference is referred to as the *prediction gain*

$$\mathrm{SNR}_p - \mathrm{SNR}_o = 10 \cdot \log_{10} \frac{D_o}{D_p} \approx 10 \cdot \log_{10} \frac{\sigma_x^2}{\sigma_d^2}$$

Estimating the auto correlation function

Given a long sequence $x_1, x_2, \ldots, x_n$ of test data, the auto correlation function can be estimated as

$$R_{XX}(k) = \frac{1}{n-k} \sum_{i=1}^{n-k} x_i \cdot x_{i+k}$$

In Matlab this can be written as

```
mean(x(1:end-k).*x(k+1:end))
```

Signals with nonzero mean

What do we do if we have a signal with a mean value $m \neq 0$?

1. If the signal mean value is small compared to the variance we can just use linear prediction.
2. If not, we can create a new signal $y_n = x_n - m$, construct a linear predictor for y_n and send m as side information.
3. Alternatively we can construct an *affine predictor*

$$p_n = \sum_{i=1}^{N} a_i x_{n-i} + a_0$$

   Disregarding the quantization this will give the same result as alternative 2.

Multidimensional predictors

We can of course generalize linear prediction to work with multidimensional signals, like images.
For example, if we have an image signal x_{ij} and want to do prediction from the pixel to the left of and the pixel above the current pixel

$$p_{ij} = a_1 \cdot x_{i,j-1} + a_2 \cdot x_{i-1,j}$$

The optimal predictor is then given by the solution to the equation system

$$\left[\begin{array}{cc} E\{x_{i,j-1}^2\} & E\{x_{i,j-1} \cdot x_{i-1,j}\} \\ E\{x_{i,j-1} \cdot x_{i-1,j}\} & E\{x_{i-1,j}^2\} \end{array} \right] \left[\begin{array}{c} a_1 \\ a_2 \end{array} \right] = \left[\begin{array}{c} E\{x_{i,j} \cdot x_{i,j-1}\} \\ E\{x_{ij} \cdot x_{i-1,j}\} \end{array} \right]$$

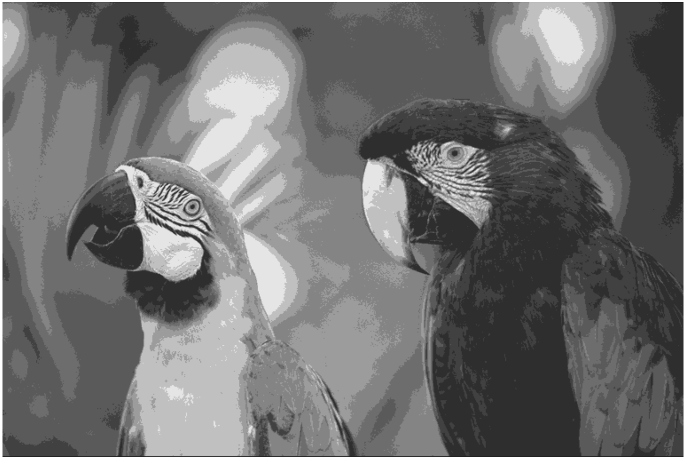
or, expressed using the auto correlation function

$$\left[\begin{array}{cc} R_{XX}(0,0) & R_{XX}(1,-1) \\ R_{XX}(1,-1) & R_{XX}(0,0) \end{array} \right] \left[\begin{array}{c} a_1 \\ a_2 \end{array} \right] = \left[\begin{array}{c} R_{XX}(0,1) \\ R_{XX}(1,0) \end{array} \right]$$

Example, predictive coding of image

Original image, 768 $\times$ 512 pixels, 8 bits/pixel

Lloyd-Max quantization, 8 levels



Rate: $R = 3$ bits/pixel
Distortion: $D \approx 59.02$
PSNR: 30.42 dB

Predictor

Estimated auto correlation function (mean removed from image)

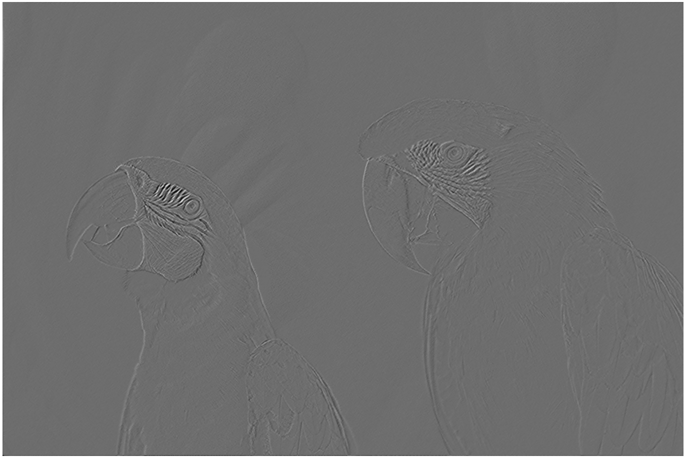
$$\begin{aligned}
R_{XX}(0,0) &= \sigma^2 \approx 2580.9 \\
R_{XX}(1,0) &\approx 0.9770 \cdot \sigma^2 \\
R_{XX}(0,1) &\approx 0.9863 \cdot \sigma^2 \\
R_{XX}(1,1) &\approx 0.9703 \cdot \sigma^2 \\
R_{XX}(1,-1) &\approx 0.9665 \cdot \sigma^2
\end{aligned}$$

Predictor

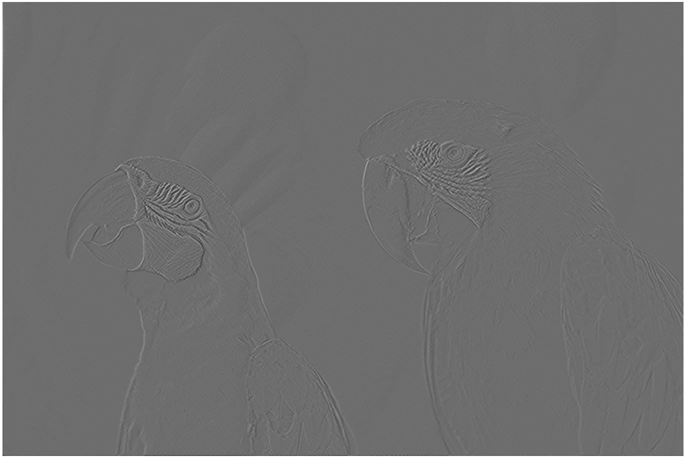
$$p_{ij} = 0.8008 \cdot \hat{x}_{i,j-1} + 0.6493 \cdot \hat{x}_{i-1,j} - 0.4525 \cdot \hat{x}_{i-1,j-1}$$

An eight level Lloyd-Max quantizer is optimized for the prediction error.

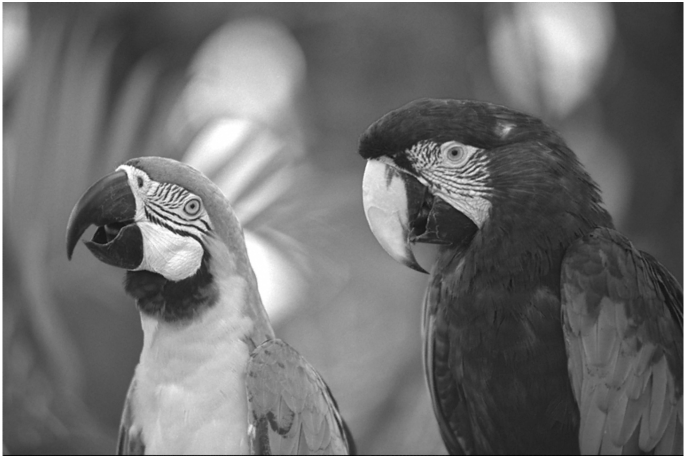
Prediction error, 8 levels



Quantized prediction error, 8 levels

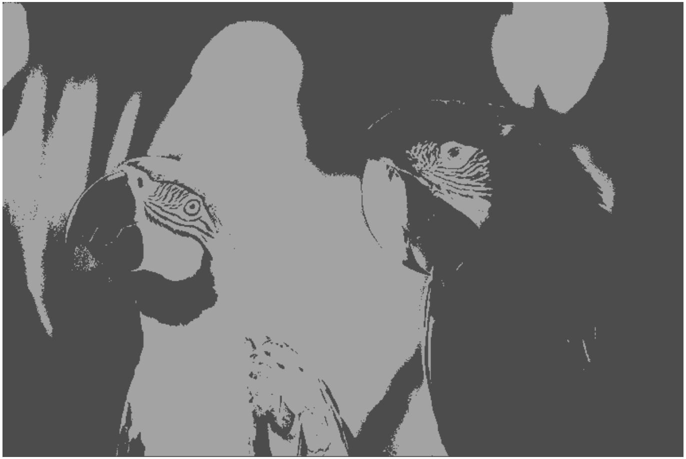


Decoded image, 8 levels



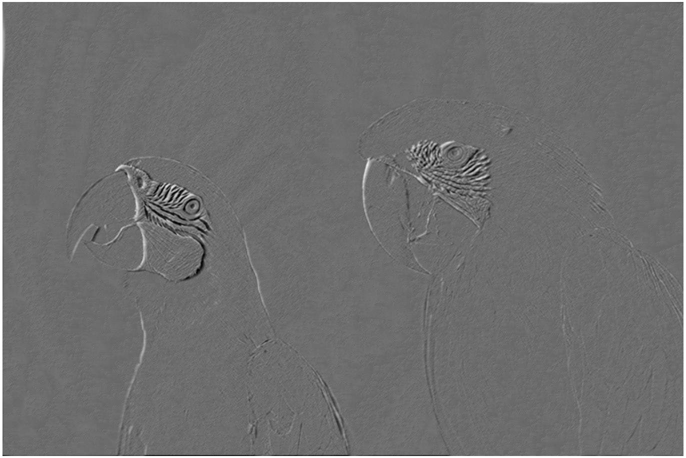
Rate: $R = 3$ bits/pixel
Distortion: $D \approx 5.62$
PSNR: 40.63 dB (Prediction gain 10.21 dB)

Lloyd-Max quantization, 2 levels

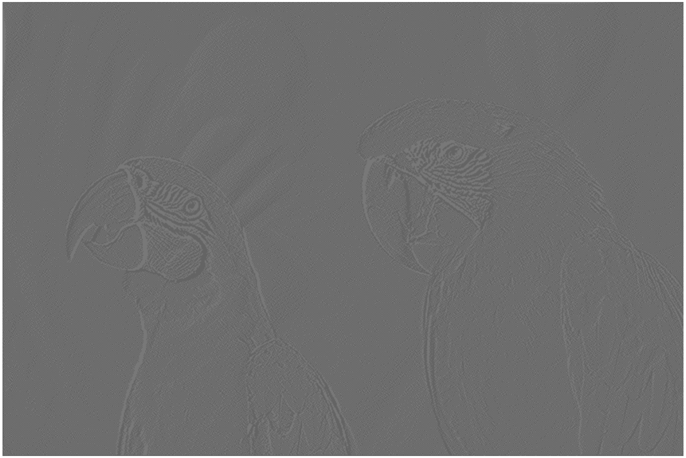


Rate: $R = 1$ bits/pixel
Distortion: $D \approx 735.77$
PSNR: 19.46 dB

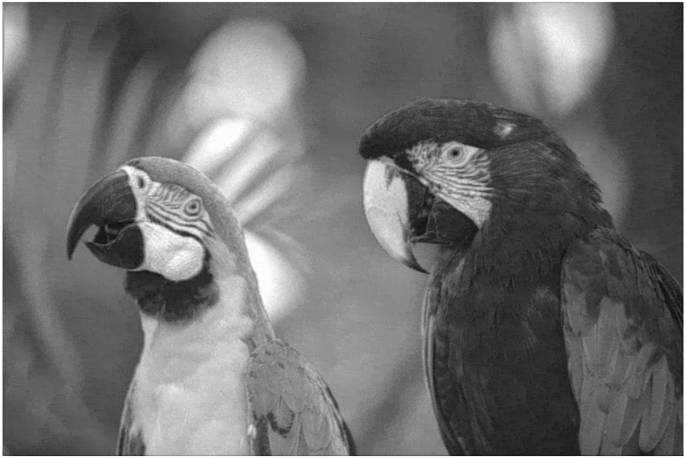
Prediction error, 2 levels



Quantized prediction error, 2 levels



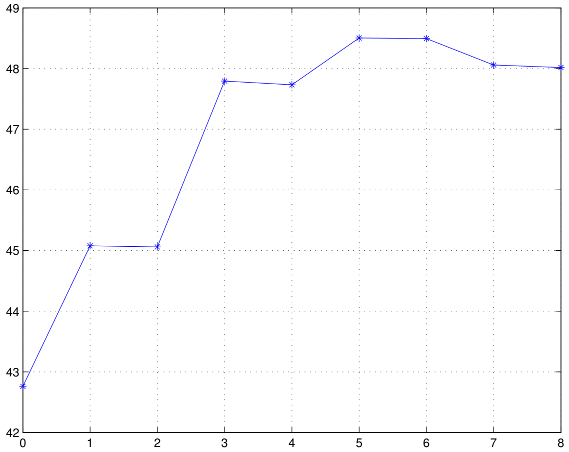
Decoded image, 2 levels



Rate: $R = 1$ bits/pixel
Distortion: $D \approx 84.81$
PSNR: 28.85 dB (Prediction gain 9.39 dB)

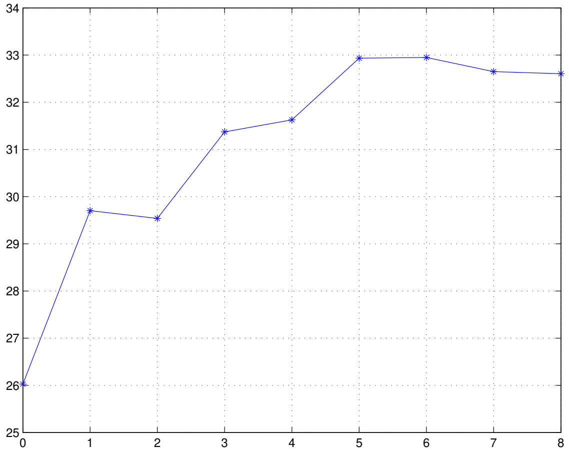
Example: `hey04.wav`

A 16 bit version of `hey04.wav` from lab 1/2 is coded using different orders of predictors.
Uniform quantization to 256 levels.
The plot shows SNR as a function of the number of predictor coefficients.



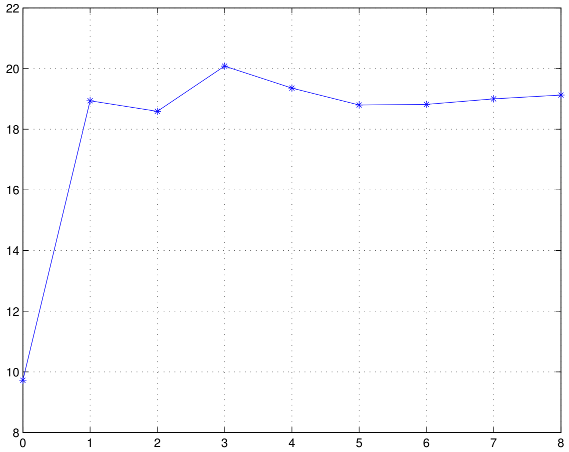
Example: `hey04.wav`

A 16 bit version of `hey04.wav` from lab 1/2 is coded using different orders of predictors.
Uniform quantization to 32 levels.
The plot shows SNR as a function of the number of predictor coefficients.



Example: `hey04.wav`

A 16 bit version of `hey04.wav` from lab 1/2 is coded using different orders of predictors.
Uniform quantization to 4 levels.
The plot shows SNR as a function of the number of predictor coefficients.



Lossless coding

Linear predictive coding can also be used for lossless coding.
Assuming that we have an integer signal, we must make sure that the predictor also produces integers.
For example we have lossless JPEG, which can use the predictors

1. $p_{ij} = I_{i-1,j}$
2. $p_{ij} = I_{i,j-1}$
3. $p_{ij} = I_{i-1,j-1}$
4. $p_{ij} = I_{i,j-1} + I_{i-1,j} - I_{i-1,j-1}$
5. $p_{ij} = \lfloor I_{i,j-1} + (I_{i-1,j} - I_{i-1,j-1})/2 \rfloor$
6. $p_{ij} = \lfloor I_{i-1,j} + (I_{i,j-1} - I_{i-1,j-1})/2 \rfloor$
7. $p_{ij} = \lfloor (I_{i,j-1} + I_{i-1,j})/2 \rfloor$

Lossless coding

We code the parrot image using the predictor

$$p_{ij} = I_{i,j-1} + I_{i-1,j} - I_{i-1,j-1}$$

followed by Huffman coding of the prediction error. We get the rate 4.18 bits/pixel.

If we instead use the predictor

$$p_{ij} = \lfloor 0.8008 \cdot I_{i,j-1} + 0.6493 \cdot I_{i-1,j} - 0.4525 \cdot I_{i-1,j-1} \rfloor$$

followed by Huffman coding we get the rate 3.93 bits/pixel.

Audio signals from labs 1 and 2

Predictors of order 1 and 2 that minimize prediction error variances.

`hey04_8bit.wav`

$$\begin{aligned} p_n &= 0.9820 \cdot x_{n-1} \\ p_n &= 1.2970 \cdot x_{n-1} - 0.3207 \cdot x_{n-2} \end{aligned}$$

`nuit04_8bit.wav`

$$\begin{aligned} p_n &= 0.9981 \cdot x_{n-1} \\ p_n &= 1.8434 \cdot x_{n-1} - 0.8468 \cdot x_{n-2} \end{aligned}$$

`speech.wav`

$$\begin{aligned} p_n &= 0.9507 \cdot x_{n-1} \\ p_n &= 1.7719 \cdot x_{n-1} - 0.8639 \cdot x_{n-2} \end{aligned}$$

FLAC (Free Lossless Audio Coding)

Lossless coding of audio signals.

The audio signal is split into blocks (typically a couple of thousand samples each).

Code the sum/difference of the two stereo channels if this gives a higher compression.

Linear predictors are optimized for each block. There is also a possibility to use fixed predictors (compare to lossless JPEG).

The prediction error is coded using Rice codes (roughly the same thing as Golomb codes).

`https://xiph.org/flac/`