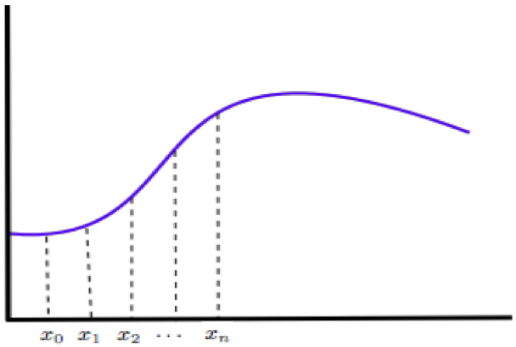
EE5585 Data Compression April 18, 2013

Lecture 23

Instructor: Arya Mazumdar *Scribe: Trevor Webster*

Differential Encoding

Suppose we have a signal that is slowly varying. For instance, if we were looking at a video frame by frame we would see that only a few pixels are changing between subsequent frames. In this case, rather than encoding the signal as is, we would first sample it and look at the difference signal and encode this instead:



$$d_n = x_n - x_{n-1}$$

This d_n would then need to be quantized, creating an estimate ($\hat{d_n}$) which would contain some quantization noise (q_n)

$$Q(d_n) = \hat{d_n}$$

$$\hat{d}_n = d_n + q_n$$

What we are truly interested in recovering are the values of x. Unfortunately, the quantization error will get accumulated in the value of x. The reason being, that the operation forming d_n is of the following matrix form

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ \vdots & \vdots & 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

Inverting this matrix we obtain the accumulator matrix

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 1 & 1 & 1 & 1 & \dots & 1 \end{pmatrix}$$

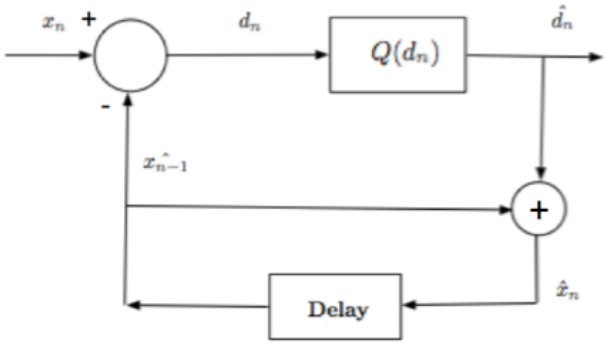
If you make an error in the d_n the noise in x_n will be accumulated as follows

$$\hat{x}_n = x_0 + \sum_{i=0}^{n} q_n$$

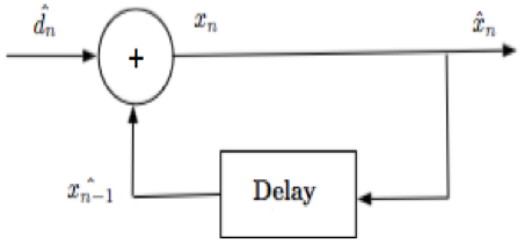
The quantization noise q_n can be positive or negative and in the long term we expect it to add up to zero, but what happens with high probability is that the range of the error becomes too great for us to handle. Therefore, we adopt the following strategy

$$d_n = x_n - \hat{x}_{n-1}$$

We can then implement this technique using the following encoder



The corresponding decoder will look as follows



This encoder/decoder scheme shown is governed by the relationship introduced previously. Rearranged it looks as follows

$$\hat{x_n} = \hat{d_n} + \hat{x}_{n-1}$$

Walking through an example using this scheme we would have the following sequence

$$\begin{aligned}
\hat{x_0} &= x_0 && (1)\\
\hat{x_1} &= \hat{d_1} + \hat{x_0} && (2)\\
&= d_1 + q_1 + \hat{x_0} && (3)\\
&= x_1 + q_1 && (4)\\
\hat{x_2} &= \hat{d_2} + \hat{x_1} && (5)\\
&= d_2 + q_2 + \hat{x_1} && (6)\\
&= x_2 + q_2 && (7)\\
& && (8)
\end{aligned}$$

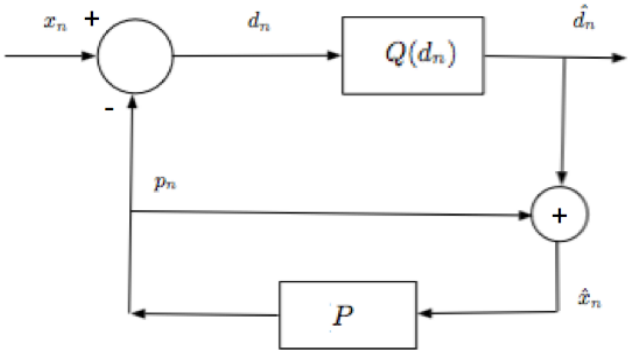
We see that in general

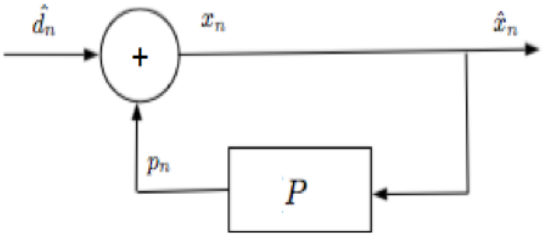
$$\begin{aligned}\hat{x_n} &= \hat{d_n} + \hat{x}_{n-1} \\ &= d_n + q_n + \hat{x}_{n-1} \\ &= x_n + q_n\end{aligned}$$

Notice that while the quantization noise q_n was originally accumulated in x_n, by adopting the aforementioned strategy we have made each estimate of x_n dependent on only its own respective quantization error. This type of method is known as a Differential Pulse Coded Modulation Scheme (DPCM).

Differential Pulse Coded Modulation Scheme

Generalizing the method described above, we see that we use a previous predicted value of x_n, denoted by p_n, to construct the difference sequence which is then quantized and used to predict the current value of x_n. Furthermore, the predicted x_n is operated on by a Predictor (P) which provides an estimate of the previous x_n in order to recursively find the difference signal. This is shown in the respective encoder and decoder figures below





Formalizing this procedure, we write

$$\begin{aligned}p_n &= f(\hat{x}_{n-1}, \hat{x_{n-2}}, \hat{x_{n-3}}, \ldots, \hat{x_0}) \\ d_n &= x_n - p_n \\ &= x_n - f(\hat{x}_{n-1}, \hat{x_{n-2}}, \hat{x_{n-3}}, \ldots, \hat{x_0})\end{aligned}$$

We are hopeful that the values of d_n are much smaller than the values of x_n, given that it is a difference signal for a slowly varying signal. So if we were looking at the energy in d_n we would guess it to be much smaller than the energy in x_n. When we are dealing with the energy of a signal it is analogous to its variance, which we define as follows
So if we were to optimize something in this DPCM scheme, we would want to find f such that we minimize the energy in d_n

Find $f(\hat{x}_{n-1}, \hat{x_{n-2}}, \hat{x_{n-3}}, \ldots, \hat{x_0})$ such that ${\sigma_d}^2$ is minimized.

$$ {\sigma_d}^2 = E[(x_n - f(\hat{x}_{n-1}, \hat{x_{n-2}}, \hat{x_{n-3}}, \ldots, \hat{x_0}))^2] $$

Since it is necessary to know previous estimates of x_n in order to calculate f, but we also need to know f in order to calculate previous estimates, we find ourselves in a roundabout and difficult situation. Therefore, we make the following assumption. Suppose,

$$ p_n = f(x_{n-1}, x_{n-2}, x_{n-3}, \ldots, x_0) $$

In other words, we design the predictor assuming there is no quantization. This is called a *Fine Quantization* assumption. In many cases this makes sense because the estimate of x_n is very close to x_n, as the difference signal is very small, and the quantization error is even smaller. Now we have

$$ {\sigma_d}^2 = E[(x_n - f(x_{n-1}, x_{n-2}, x_{n-3}, \ldots, x_0)^2] $$

Considering the limitations on f, we note that f can be any nonlinear function. But we often don't know a way to implement many nonlinear functions. So we assume further that the predictor f is linear and seek the best linear function. By definition, a linear function expressing $f(x_{n-1}, x_{n-2}, x_{n-3}, \ldots, x_0)$ will merely be a weighted sum of previous values of x_n. Assuming we are looking back through a "window" at the first N samples of a signal, the predictor can then be expressed as follows

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

Now the variance of d_n becomes

$$ {\sigma_d}^2 = E[(x_n - \sum_{i=1}^{N} a_i x_{n-i})^2] $$

Now we need to find the coefficients a_i which will minimize this variance. To do this we take the derivative with respect to a_i and set it equal to zero.

$$ \begin{aligned} \tfrac{\partial {\sigma_d}^2}{\partial a_1} &= E[-2(x_n - \textstyle\sum_{i=1}^{N} a_i x_{n-i}) x_{n-1}] \\ &= -2E[x_n x_{n-1} - \textstyle\sum_{i=1}^{N} a_i x_{n-i} x_{n-1}] = 0 \\ &\vdots \\ \tfrac{\partial {\sigma_d}^2}{\partial a_j} &= 2E[x_n x_{n-j} - \textstyle\sum_{i=1}^{N} a_i x_{n-i} x_{n-j}] = 0 \end{aligned} $$

By setting the derivative above to zero and rearranging, we see that the coefficients we are looking for are dependent upon second order statistics

$$ E[x_n x_{n-j}] = \sum_{i=1}^{N} a_i E[x_{n-i} x_{n-j}] $$

We make the assumption that x is stationary, wich means that the correlation between two values of x_n is only a function of the lag between them. Formally, we denote this by the fact that the expectation of the product of two values of x_n separated in time by k samples (also known as the *autocorrelation*) is purely a function of the time difference, or lag, k

$$ E[x_n x_{n+k}] = R_{xx}(k) $$

The relationship governing the coefficients a_i can then be rewritten using this notation as

$$ R_{xx}(j) = \sum_{i=1}^{N} a_i R_{xx}(i-j) \text{ for } 1 \leq j \leq N $$

Thus, looking at the autocorrelation functions for each j we have

$$R_{xx}(1) = \sum_{i=1}^{N} a_i R_{xx}(i-1)$$
$$R_{xx}(2) = \sum_{i=1}^{N} a_i R_{xx}(i-2)$$
$$\vdots$$
$$R_{xx}(N) = \sum_{i=1}^{N} a_i R_{xx}(i-N)$$

Now that we have N equations and N unknowns we can solve for the coefficients a_i in matrix form

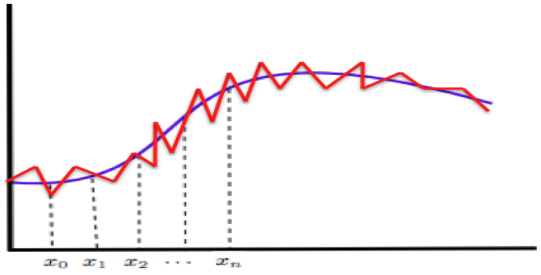
$$\begin{pmatrix} R_{xx}(0) & R_{xx}(1) & \dots & R_{xx}(N-1) \\ R_{xx}(1) & R_{xx}(0) & & \vdots \\ \vdots & & \ddots & \vdots \\ R_{xx}(N-1) & & \dots & R_{xx}(0) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} R_{xx}(1) \\ R_{xx}(2) \\ \vdots \\ R_{xx}(N) \end{pmatrix}$$

Rewriting this more compactly where R is a matrix and a and P are column vectors we have

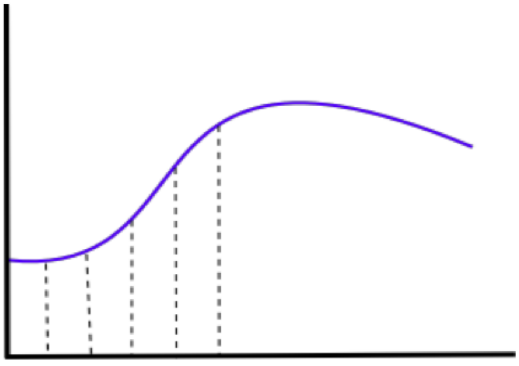
$$Ra = P$$
$$a = R^{-1}P$$

Thus, we see in order to determine the coefficients for a linear predictor for DPCM we must invert the autocorrelation matrix and multiply by the autocorrelation vector P. Once we have determined the coefficients we can design the predictor necessary for our DPCM scheme.

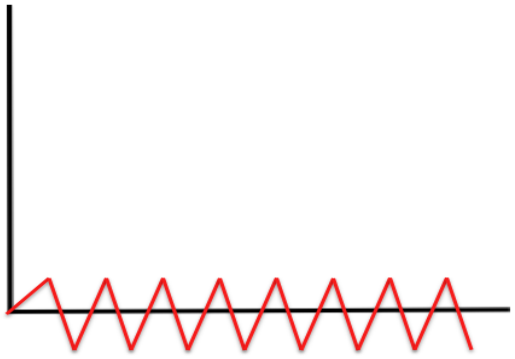
Now suppose that our signal is rapidly changing as shown below



We see that since the signal varies significantly over short intervals the difference signal d_n would be very large and DPCM might not be a very good scheme. Suppose instead that we passed this rapidly changing signal through both a low pass and high pass filter as shown below



Low Pass



High Pass

Let us create two signals based off the values of x_n to emulate this low pass and high pass response. Let y_n represent an averaging operation that smoothes out the response of x_n (low pass) and let z_n represent a difference operation which emulates the high frequency variation of x_n (high pass).

$$y_n = \frac{x_n + x_{n-1}}{2}$$
$$z_n = \frac{x_n - x_{n-1}}{2}$$

Applying this method for each x_n we would send or store two values (y_n and z_n). This is unnecessary. Instead, what we can do is apply the following strategy

$$y_{2n} = \frac{x_{2n} + x_{2n-1}}{2}$$
$$z_{2n} = \frac{x_{2n} - x_{2n-1}}{2}$$

Then we can recover both even and odd values of x_n as follows

$$y_{2n} + z_{2n} = x_{2n}$$
$$y_{2n} - z_{2n} = x_{2n-1}$$

This process of splitting signal components into multiple portions is called decimation and can be extrapolated out further until a point at which you perform bit allocation. This method is called sub-band coding. We note that DPCM is well suited for the y_n low pass components whereas another technique would likely suite the z_n high pass components more effectively.

Distributed Storage

Today's storage systems often utilize distributed storage. In such a system, data will be stored in many separate locations on servers. Suppose we want to do data compression in such a system. What we would like to do is query a portion of our data set - this could be 1 page out of an entire document for instance. Rather than having to sift through the entire data set to find 1 page, we would like to be able to go directly there. In other words, we would like such a system to be *query efficient*. For this reason suppose we divide data out into sections, such as by page, before compressing it in an effort to preserve information about where the data came from. Such a partitioning of a data set is shown below

$$(101|010|011|\ldots|\ldots)$$

Next we define *query efficiency* as the # of bits we need to process to get back a single bit. Suppose that the partitioned bit stream shown above has length N and we have partitioned it in chunks of $m = 3$ bits. In this instance, the query efficiency would be m.

Suppose we have a binary vector of length n represented by x which we can compress to $H(x) + 1$. Our compression would then be

$$\begin{aligned}\text{Compression Rate} &= \frac{H(x_1^n)+1}{n} \\ &= \frac{H(x_1^n)}{n} + \frac{1}{n} \\ &= \frac{nH(x)}{n} + \frac{1}{n} \\ &= H(x) + \frac{1}{n}\end{aligned}$$

Now, in the case of our previous example regarding query efficiency. The compression rate will be

$$\begin{aligned}\text{Rate} &= H(x) + \frac{1}{m} \\ &= H(x) + \frac{1}{query\ efficiency}\end{aligned}$$

If we were able to compress the file as a whole the query efficiency would be huge and the Compression Rate would be

$$\text{Rate} = H(x) + \frac{1}{N}$$
$$\text{Difference in rate from optimal} = \frac{1}{Query\ Efficiency}$$

Thus we see there is a trade off between Compression Rate and Query Efficiency. Let us end the lecture by propose a research problem were we find a better relation between query efficiency and redundancy.