EE5585 Data Compression April 4, 2013

Lecture 19

Instructor: Arya Mazumdar *Scribe: Katie Moenkhaus*

Kolmogorov Complexity

A major result is that the Kolmogorov complexity of a random sequence on average is close to the entropy. This captures the notion of compressibility. Also, an algorithmically incompressible binary sequence as defined by the Kolmogorov complexity has approximately the same numer of 1s and 0s. The difference between the number of 0s and 1s gives insight into how compressible a signal is. Kolmogrov complexity can be defined for numbers as well as sequences. For $n \in \mathbb{Z}$,

$$K(n) = \min_{p:U(p)=n} l(p)$$

where $K(n)$ denotes the Kolmogorov complexity of n, and $l(p)$ is the minimum length of the computer program. Some integers have low Kolmogorov complexity. For exmple, let $n = 5^{5^{5^{5^{5}}}}$. Even though this is a very large number, it has a short description. Also, e is easily described. Even though it is an irrational numer, it has a very short description of the basis of the natural logarithm, so it is actually a function such that the derivative of the function is itself.

Note,

$$K(n) \leq \log^* n + c$$

where $\log^* n = \log n + \log\log n + \log\log\log n + \ldots$, so long as the each term is positive.

Theorem 1 *There exists an infinite number of integers for which $K(n) > \log n$.*

Proof (by contradiction): Assume there exist a finite number of integers for which $K(n) > \log n$. From Kraft's Inequality,

$$\sum_n 2^{-K(n)} \leq 1$$

If the assumption is true, then there will be only a finite number of integers for which $K(n) > \log n$. Then there is a number n_0 for which all $n \geq n_0 \implies K(n) \leq \log n$. Then

$$\sum_{n \geq n_0} 2^{-K(n)} \geq \sum_{n \geq n_0} 2^{-\log n} = \sum_{n \geq n_o} \frac{1}{n}$$

The series $\sum_n \frac{1}{n}$ doesn't converge, so

$$\sum_{n \geq n_o} \frac{1}{n} = \infty$$

It follows that

$$\sum_n 2^{-K(n)} = \infty$$

This contradicts with $\sum_n 2^{-K(n)} \leq 1$, and thus the assumption is false. $\therefore \exists$ a finite number of integers for which $K(n) > \log n$. ■

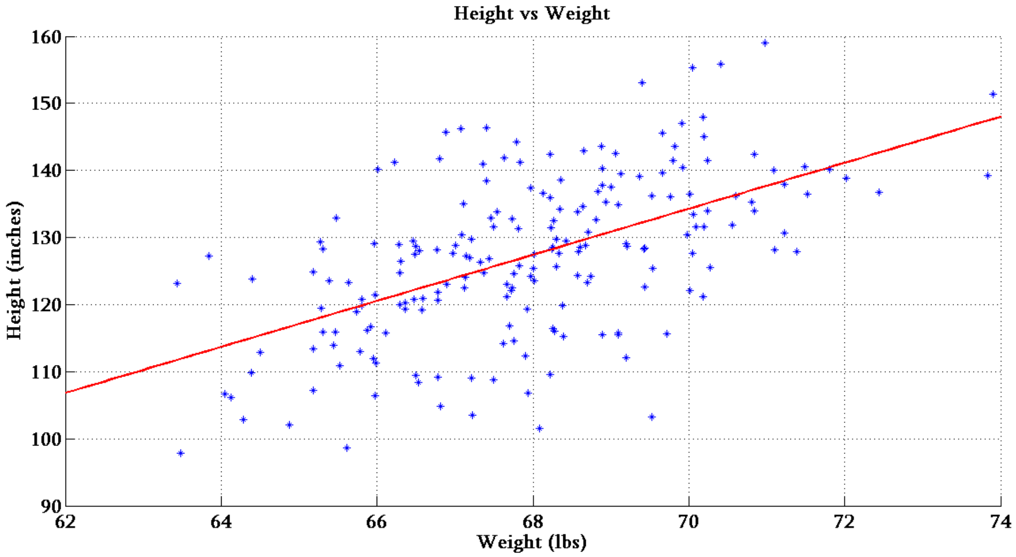
This illustrates that there is an infinite number of integers that are not simple to describe, i.e. they will take more than $\log n$ bits to describe. Given a binary sequency, if

$$|\#1s - \#0s| \geq \epsilon_n$$

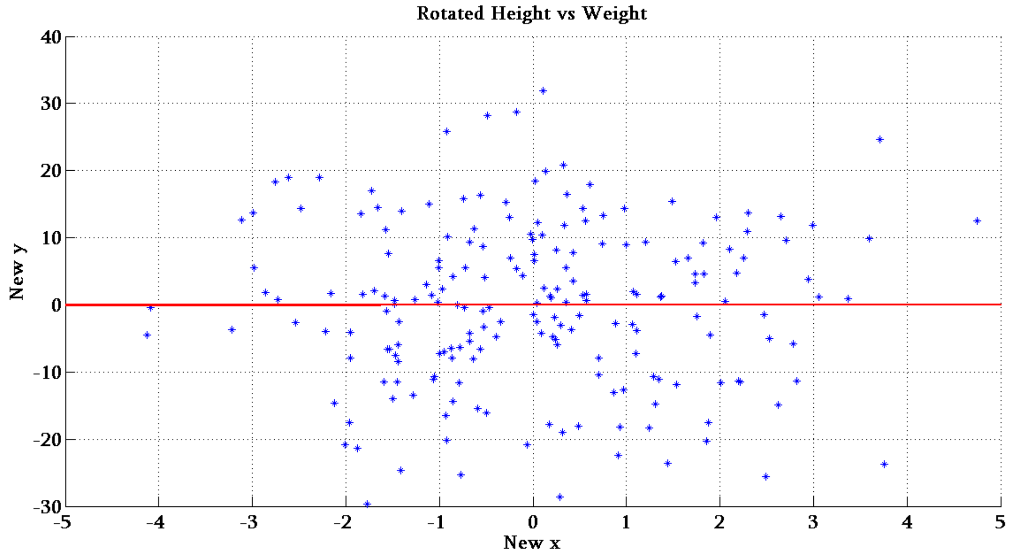
then the sequence can be compressed.

Transform Coding

Consider a height/weight data set.



Because the data is highly correlated, a clockwise rotation can be done to obtain new axes such that the line of best fit becomes the new prominent axis.



Thus the difference between any data point and the axis is small, and fewer bits are needed to describe them. In the new data set, the entries are uncorrelated. Ideally, there will be 0 correlation in the new coordinates. Note that the correlation between random variables X and Y is $E[(X - E(X))\,(Y - E(Y))]$, where $E(X)$ denotes the expected value of X. If X and Y are independent, then $Cor = 0$. Similarly, if the correlation is large, the variables are highly correlated. Without loss of generality, $E(X) = E(Y) = 0$.

This is just subtracting the mean from each set. Let

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}; \quad E(X) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

In the height/weight data, there were only two variables, i.e. height is x_1 and weight is x_2. The rotation is done by multiplying by a matrix:

$$Y = AX$$

Y is another N-dimensional vector, and A is $N \times N$. It is desirable to have A be an orthonormal matrix, i.e. $A^T A = I$. If this holds, then the result is an orthogonal transformation. For any orthogonal transformation, Parseval's Theorem is true. This says that

$$\begin{aligned} \sum_i \sigma_{y_i}^2 &= \sum_i E(y_i^2) \\ &= E(\|Y\|_2^2) \\ &= E(Y^T Y) \\ &= E(X^T A^T A X) \\ &= E(X^T X) \\ &= E(\|X\|_2^2) \\ &= \sum_i E(x_i^2) = \sum_i \sigma_{x_i}^2 \end{aligned}$$

If σ_x is the variance of x, then $\sum_i E(x_i^2) = \sum_i \sigma_{x_i}^2$ stems from the fact that X is a zero-mean, random variable. Thus, the transformation conserves the total energy. The transformation should ensure that the new data are uncorrelated. The correlation is representated by the covariance matrix, C_x, defined as

$$\begin{aligned} C_x = E(XX^T) &= E \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \dots & x_N \end{bmatrix} \\ &= E \begin{bmatrix} x_1^2 & x_1 x_2 & \dots & x_1 x_N \\ x_1 x_2 & x_2^2 & \dots & x_2 x_N \\ \vdots & \vdots & \ddots & \vdots \\ x_1 x_N & x_2 x_N & \dots & x_N^2 \end{bmatrix} \\ &= \begin{bmatrix} \sigma_{x_1}^2 & E(x_1 x_2) & \dots & E(x_1 x_N) \\ E(x_1 x_2) & \sigma_{x_2}^2 & \dots & E(x_2 x_N) \\ \vdots & \vdots & \ddots & \vdots \\ E(x_1 x_N) & E(x_2 x_N) & \dots & \sigma_{x_N}^2 \end{bmatrix} \end{aligned}$$

Note that this is a symmetric matrix. From the definition of the trace of a matrix,

$$Trace(C_x) = \sum_i \sigma_{x_i}^2$$

which is the total energy of the signal. The transform matrix A should minimize the off-diagonal elements of the covariance matrix.

$$\begin{aligned} C_y &= E(YY^T) \\ &= E(AXX^TA^T) \\ &= AE(XX^T)A^T \\ &= AC_xA^T \end{aligned}$$

Because $A^TA = I$ and A is an orthogonal matrix, $A^T = A^{-1}$. Thus,

$$\begin{aligned} C_y &= AC_xA^{-1} \\ C_yA &= AC_x \end{aligned}$$

The ideal covariance matrix C_y is a diagonal matrix. Suppose

$$C_y = \begin{bmatrix} \lambda_1 & & & & \text{\huge 0} \\ & \lambda_2 & & & \\ & & \ddots & & \\ \text{\huge 0} & & & \lambda_{N-1} & \\ & & & & \lambda_N \end{bmatrix}$$

Thus

$$\begin{bmatrix} \lambda_1 & & & & \text{\huge 0} \\ & \lambda_2 & & & \\ & & \ddots & & \\ \text{\huge 0} & & & \lambda_{N-1} & \\ & & & & \lambda_N \end{bmatrix} A = AC_x$$

Because X is zero mean, Y is also zero mean. Suppose

$$\begin{aligned} A &= \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{bmatrix} \\ \Longrightarrow C_yA &= \begin{bmatrix} \lambda_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{N1} \end{bmatrix} & \lambda_2 \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{N2} \end{bmatrix} & \dots & \lambda_N \begin{bmatrix} a_{1N} \\ a_{2N} \\ \vdots \\ a_{NN} \end{bmatrix} \end{bmatrix} \\ &= \begin{bmatrix} C_x \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{N1} \end{bmatrix} & C_x \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{N2} \end{bmatrix} & \dots & C_x \begin{bmatrix} a_{1N} \\ a_{2N} \\ \vdots \\ a_{NN} \end{bmatrix} \end{bmatrix} \end{aligned}$$

For ease of notation, let

$$A_i = \begin{bmatrix} a_{1i} \\ a_{2i} \\ \vdots \\ a_{Ni} \end{bmatrix}$$

In general,

$$C_x A_i = \lambda_i A_i$$

λ_i is an eigenvalue of C_x, and A_i is the corresponding eigenvector.

$$A = \begin{bmatrix} A_1 & A2 & \dots & A_N \end{bmatrix}$$

Thus, the transform matrix A is

$$A = \begin{bmatrix} eigenvectors \;\; of \;\; C_x \end{bmatrix}$$

The eigenvalues are simply $\lambda_i = \sigma_{y_i}^2$. This is called the **Karhunen-Loève Transform** (KLT). The process for finding the KLT is

1. Subtract off the mean:
   $x_1 \to x_1 - E(x_1)$
   $x_2 \to x_2 - E(x_2)$
   $\vdots$
   $x_N \to x_N - E(x_N)$

2. Find the covariance matrix. If there are M elements in each vector, then the total energy is

   $$\sigma_{x_n}^2 = \frac{1}{M} \sum_{i=1}^{M} x_{ni}^2$$

   for each n going from 1 to N. The correlations are then

   $$\rho_{x_n, x_m} = \frac{1}{M} \sum_{i=1}^{M} x_n x_m$$

   The covariance matrix is then

   $$C_x = \begin{bmatrix} \sigma_{x_1}^2 & \rho_{x_1,x_2} & \cdots & \rho_{x_1,x_N} \\ \rho_{x_1,x_2} & \sigma_{x_2}^2 & \cdots & \rho_{x_2,x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{x_1,x_N} & \rho_{x_2,x_N} & \cdots & \sigma_{x_N}^2 \end{bmatrix}$$

3. Knowing the covariance matrix C_x, find the eigenvalues and eigenvectors. Form the transform matrix A as

   $$A = \begin{bmatrix} A_1 & A2 & \dots & A_N \end{bmatrix}$$

   where A_i represents the i^{th} eigenvector of C_x.

The process outlined above is called **Principal Component Analysis**. The goal of this analysis is to find a matrix that transforms the vectors to a new coordinate system in which they are uncorrelated. So ideally,

$$C_y = \begin{bmatrix} \lambda_1 & & & & \\ & \lambda_2 & & \text{\Large 0} & \\ & & \ddots & & \\ & \text{\Large 0} & & \lambda_{N-1} & \\ & & & & \lambda_N \end{bmatrix}$$

Keep only the eigenvalues that are largest. The energy lost in this process is the sum of the eigenvalues that are not used in the compression scheme. This is the mean square error. Suppose that the eigenvalues are ordered, i.e. $\lambda_1 > \lambda_2 > \cdots > \lambda_N$. If all $\{\lambda_i\}_{i=N_0}^{N}$ are unused, then the mean square error is

$$MSE = \sum_{i=N_0}^{N} \lambda_i$$

This is one measure of the distortion induced by compressing the signal. KLT optimally minimizes $E(y_i y_j)$, but it is computationally inefficient because for each new data set, a new transform matrix has to be computed. Instead, some standard transform matrices are used. One such of these is

$$F = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \dots & \omega^N \\ 1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{2N} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \dots & \omega^{3N} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \omega^{3(N-1)} & \dots & \omega^{(N-1)(N-1)} \end{bmatrix}$$

This is an example of a generic transform matrix. Typically, $\omega = e^{-i\frac{2\pi}{N}}$. This is the **Discrete Fourier Transform** (DFT) matrix. The DFT projects data along the rows, and each row has a frequency kernel. Thus, it produces the frequency components of the data.

$$\begin{aligned} Y &= FX \\ &= \begin{bmatrix} Y1 \\ Y2 \\ \vdots \\ Y_N \end{bmatrix} \end{aligned}$$

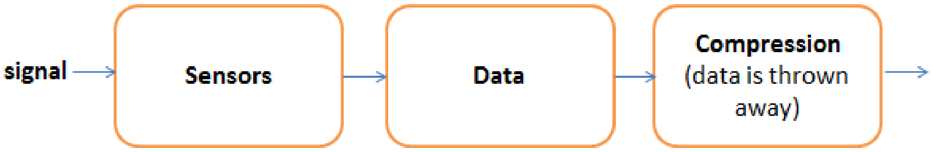
The values $\{Y_i\}_{i=1}^{N}$ are the frequency components. Compressing a signal by omitting any of the low amplitude $Y_i s$ (usually high frequency) effectively filters the signal.

If the low amplitude $Y_i s$ are scattered about the vector, filtering is still being done, but it is not necessarily characterizable as a low, high, or band-pass filter. This is a widely used compression scheme. In the case of images, most frequently a **Discrete Cosine Transform** (DCT) is used.

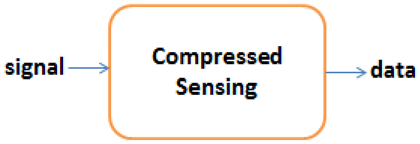
$$A_{ij} = \begin{cases} \sqrt{\frac{1}{N}}\, cos\left(\frac{j\pi(2i+1)}{2N}\right) & i = j \\ \sqrt{\frac{2}{N}}\, cos\left(\frac{j\pi(2i+1)}{2N}\right) & i \neq j \end{cases}$$

For correlated data that forms a first-order Markov chain, then the energy compaction factor is very close to the energy compaction factor of KLT. The DCT has very good performance for highly correlated data.

A problem with modern technology is that image capturing systems, for example, lack hardware capable of these data compression techniques, so much more information is captured than is actually used after the compression. The process is outlined below.



Ideally, the sensing and compression would be combined into a compressed sensing step.



The compressed sensing block contains a matrix Φ. If the discrete signal is X, ideally the sensors will be able to do the transform matrix multiplication ΦX. Given any signal to be sensed, X, there is hardware to implement the linear combination of the rows of Φ. The multiplication is

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{M1} & \phi_{M2} & \dots & \phi_{MN} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix}$$

where $M \ll N$ for compression. Note that Φ is no longer a square matrix; it is short and fat. Thus only M samples are taken, and compression and sensing occur all in the same step. This can be done in hardware.

Decoding Transform Codes

In decoding, X needs to be recovered from Y, but there is not a unique solution for X given Φ and Y. However, some more information is known about X. Given any transform matrix F, FX is a vector that has few non-zero entries. F is known.

$$\Phi = \tilde{\Phi} F$$

Note that Φ is $M \times N$, $\tilde{\Phi}$ is $M \times N$, and F is $N \times N$.

$$\begin{aligned} Y &= \Phi X \\ &= \tilde{\Phi} F X \\ &= \tilde{\Phi} \tilde{X} \end{aligned}$$

where $\tilde{X} = FX$. $\tilde{X}$ is a sparse vector, i.e. it has few non-zero values. Let k be the number of non-zero entries of $\tilde{X}$. $k \ll N$. Knowing $\tilde{\Phi}$ is equivalent to knowing Φ, since F is known. The formulation of the problem is then: Given $Y = \Phi X$ and Φ, find X, where X has $k \ll N$ non-zero values. Also find the matrix Φ for which this problem is solvable.

EE5585 Data Compression April 9, 2013

Lecture 20

Instructor: Arya Mazumdar *Scribe: Aritra Konar*

1 Review

In the last class, we studied about Transform Coding, where we try to assign a new set of basis vectors to the data. If the data exhibits some form of correlation, we apply a linear unitary transformation to rotate the axes such that most of the data fits through one of the axes. The result is that the data is sparse, and along this axis. Meaning we can throw away the entries that have values near zero, thereby compressing the data. However, this approach suffers from two drawbacks.

i) It may happen that the data cannot be fit through a single (or a smaller set of) axis. In this case the linear transforms (using DFT matrix or DCT matrix for example) will not be able to compress the data. This is shown in the diagram below, where the data is fitted by a nonlinear function.

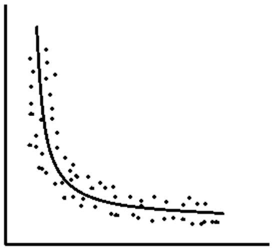


Figure 1: Example of data fitting by a non - linear function

ii) In this approach, the the data is acquired from a sensor array, following which a basis is determined in which the representation of the data is sparse, and then the sparse entries are thrown away to achieve data compression. Since a lot of the data samples obtained are thrown away, we may then seek an approach where we can compress the data by sensing only a small number of samples.

2 Compressed Sensing/Sampling

Suppose we have N pixels of an image we want to compress. We may capture the pixels using an array of N sensors to obtain a data vector, to which we then apply a linear unitary transformation (by multiplying it with the DFT matrix for example) to obtain a sparse representation of the data. We can then throw away the small entries to compress the data.

Alternatively, we may only use M ($\ll$N) sensors to capture the pixels. The diagram below illustrates this approach.

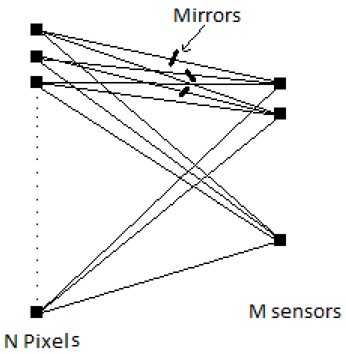


Figure 2: Single Pixel Camera

Here, each sensor senses a linear combination of N pixel values. The output of each sensor is given by

$$y_i = a_{1i}z_1 + a_{2i}z_2 + ... + a_{Ni}z_N$$

where the coefficients of the linear combination are determined from the reflection coefficients of the mirrors placed at an angle θ

Hence, the output of the sensor array consists of M linear samples drawn from z. The output vector y may be represented as

$$y = Az$$

where $A \rightarrow M \times N$, $y \rightarrow M \times 1$, $z \rightarrow N \times 1$

We observe that dimensionality reduction is achieved by mapping the $N \times 1$ vector of pixels into a $M \times 1$ vector of observations, since $M \ll N$.

Now, z has a property. We take the DFT matrix and use it to transform z into a sparse vector having only $k \ll N$ non - zero entires (i.e, z is compressible).

The next problem we face is how to recover z from the compressed data vector y. If the matrix A were square, full rank and hence invertible, we could have easily recovered z from y by the relation

$$z = A^{-1}y$$

However, since A is a fat matrix, we have an underdetermined system of linear equations with an infinte number of solutions. We will try to use the sparsity property of z(in some domain) to recover z given A, y.

Decompose

$$A = \Phi F$$

where $A \rightarrow M \times N$, $\Phi \rightarrow M \times N$, $F \rightarrow N \times N$ (Fourier Matrix in some domain where z has a sparse representation).

This is a valid matrix decomposition since F is an invertible matrix $\Longrightarrow \Phi = AF^{-1}$

Hence, we have $y = Az= \Phi Fz = \Phi x$ where $x= Fz$. x is an $N \times 1$vector having only $k \ll N$ non - zero values. Thus, in other words, x is k sparse.

We have now reduced the problem to the one of recovering x from the under-determined system of equations $y = \Phi x$ where x is k sparse. The matrix Φ is to be designed such that unambiguous recovery of x is possible. This happens if and only if, for any two k sparse vectors x_1 and x_2

$$\Phi x_1 \neq \Phi x_2$$

$$\Rightarrow \Phi(x_1 - x_2) \neq 0$$

This statement implies that any two k sparse vectors should not get mapped to the same obeservation vector y. Otherwise, unique recovery will not be possible. Since $x_1 - x_2$ is atmost $2k$ sparse, the left hand side of the above expression is the weighted sum of any $2k$ columns of the matrix Φ. Since the linear combination of the columns $\neq 0$, it follows that any $2k$ columns of Φ must be linearly independent.

$$\Rightarrow M \geq 2k$$

The above expression implies that for stable recovery, the number of samples required for compressive sensing must be greater than equal to twice the sparsity of the data vector x. Note that this statement is somewhat analogous to the Nyquist Sampling Theorem.

Also, since we do not know the locations of the k sparse samples in the $N \times 1$data vector x, we must use $M \geq 2k$ samples. If we knew the exact locations of the sparse entries, then we would require only k columns of Φ to be linearly independent, implying that only M= k samples would be sufficient for stable recovery. Hence, we see that we must pay a penalty for not knowing the locations of the sparse entries of x by requiring a greater number of samples.

Our ultimate goal, now, is to determine a matrix Φ that is $2k \times N$ dimensional and any $2k$ columns are linearly independent.

Let us consider an example of signal recovery where x is 1 sparse 6×1 vector and Φ is given by

$$\Phi = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

If the observation vector $y = \Phi x$ is also known, then we can easily recover x. We know that y is obtained from a linear combination of the columns of Φ with the coefficients of the combination being the elements of the vector x. Since x is 1 sparse, with only a single non - zero entry, then y is simply the scaled column of Φ whose coefficient is the non - zero entry of x. Hence, knowing y, we can determine the non -zero entry of x and from the position of the corresponding column in Φ, we can determine the position of the non - zero entry in x.

Note however, in this case, we used $M = 3 > 2$ samples in the sampling matrix Φ. In general, if we wanted to use $M = 2k$ samples, we could use a *Vandermonde matrix* given by

$$\Phi = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ \alpha_1 & \alpha_2 & . & . & . & \alpha_N \\ \alpha_{1^2} & \alpha_{2^2} & . & . & . & \alpha_{N^2} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \alpha_{1^{N-1}} & \alpha_{2^{N-1}} & . & . & . & \alpha_{N^{N-1}} \end{bmatrix}$$

where $\alpha_i \in \mathbb{R}$, such that $\alpha_i \neq \alpha_j \; \forall i \neq j$

We can choose any $M \times M = 2k \times 2k$ submatrix of Φ which will be non - singular, full rank with m linearly independent columns.

$$\Psi = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ \alpha_{i_1} & \alpha_{i_2} & . & . & . & \alpha_{i_M} \\ \alpha_{i_1}^2 & \alpha_{i_2}^2 & . & . & . & \alpha_{i_M}^2 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \alpha_{i_1}^{M-1} & \alpha_{i_2}^{M-1} & . & . & . & \alpha_{i_M}^{M-1} \end{bmatrix}$$

where we choose $i_1, i_2, ..., i_M$from the matrix Φ

Let $\alpha_{i_j} = \beta_j$. Then,

$$\Psi = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ \beta_1 & \beta_2 & . & . & . & \beta_M \\ \beta_1^2 & \beta_2^2 & . & . & . & \beta_M^2 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ \beta_1^{M-1} & \beta_2^{M-1} & . & . & . & \beta_M^{M-1} \end{bmatrix}$$

Ψ is a square Vandermonde matrix with

$$det\Psi = \prod_{1 \leq j < i \leq m} (\beta_i - \beta_j)$$

Since $\beta_i \neq \beta_j$ for $i \neq j$,Ψ is guaranteed to be non singular, full rank with M= $2k$ linearly independent columns.

Thus, the Vandermonde matrix Φ can be used to recover x from the observation y. However, the entire success of this scheme hinges on the assumption made at the beginning that x is k sparse. In reality, x may only be approximately k sparse, with k prominent values and another $N - k$ values that are close to, but not zero. If this is the case, then this scheme fails, because only $M = 2k$ columns are not enough to guarantee recovery. The contributions from the other columns of will be non - zero in this case since x is not exactly k sparse. We now seek a scheme that guarentees stable recovery even if x is approximately k sparse.

Suppose $\hat{x}$ is our estimate of x. The error ε is given by

$$\varepsilon = \parallel \hat{x} - x \parallel_{l_2}^2 = (\hat{x} - x)^T (\hat{x} - x)$$

Now, we would like to bound this error even if x is not exactly k sparse.

Suppose x_k is a vector that has the k largest co-ordinates of x with all others zero. Then, we may write an approximately k sparse vector x in terms of x_k as

$$x = x_k + \varepsilon$$

$$\Longrightarrow \varepsilon = x - x_k$$

where x is an exact k sparse vector and ε is the error in x not being exactly k sparse.

We would like a recovery guarantee of the form

$$\parallel \hat{x} - x \parallel_{l_2} \leq c \parallel x - x_k \parallel_{l_2} \text{........}(\nabla)$$

The above statement implies that when x is exactly k sparse, then $x_k = x$ and the right hand side of the above equation is zero, from which we get $\hat{x} = x$. In this case, perfect recovery is achieved. However, when x is only approximately k sparse, even then $\hat{x}$ is not too far from x! We are now ready to state a theorem which gives the recovery guarantee given by (∇) subject to some conditions.

3 Theorem

Recovery with guarantee given by (∇) will require $M = c_1 k \log(N/k)$ samples. Then, there exists a $M \times N$ sampling matrix Φ with $M = c_1 k \log(N/k)$ which gives guarantee

$$\parallel \hat{x} - x \parallel_{l_2} \leq \frac{c}{\sqrt{k}} \parallel x - x_k \parallel_{l_1} \text{........}(\$)$$

Note that, since we are dividing by $\sqrt{k}$, the error bound obtained is quite tight. This scheme guarantees stable recovery even when x is only approxumately k sparse. We can determine $\hat{x}$ which satisfies the above error bound by a linear program called *Basis Pursuit* which is a polynomial time algorithm.

3.1 Basis Pursuit

Given $y = \Phi x$ where x is approximately k sparse, then we can obtain $\hat{x}$ which gives guarantee (\$) by solving the following optimization problem with a linear constraint

$$\min_{s.t. \Phi \hat{x} = y} \parallel \hat{x} \parallel_{l_1}$$

3.2 Fun Facts about Norms

If q is an integer, then the L^q norm of a $N \times 1$ vector x is given by

$$\parallel x \parallel_{l_q} = (\sum_{i=1}^{N} \mid x_i \mid^q)^{\frac{1}{q}}$$

When $q \to \infty$, $L^\infty = max \mid x_i \mid$
When $q \to 0$, L^0 =number of non zero entries of x

4 Restricted Isometry Property

A sampling matrix $\Phi$is said to possess δ_k RIP, if, for any k sparse vector v,

$$(1 - \delta_k) \|v \parallel_2^2 \leq \parallel \Phi v \parallel_2^2 \leq (1 + \delta_k) \parallel v \parallel_2^2$$

We choose the smallest δ_k such that the above expression is true. Note that, if δ_k is 0, then

$$\parallel v \parallel_2^2 = \parallel \Phi v \parallel_2^2$$

Thus, Φ is an orthogonal transform that preserves the norm (energy) of the k sparse vector v.However, even if $\delta_k \neq 0$, even then Φ is approximately orthogonal since the energy in the signal is only changed by a factor δ_k. In 2005, Candes and Tao showed that if

$$\delta_{2k} \leq \sqrt{2} - 1$$

That is, if for any $2k$ sparse vector the energy is preserved by a factor of $\sqrt{2} - 1 = 0.414$,then, $\hat{x}$ obtained from Basis Pursuit will give guarantee

$$\parallel \hat{x} - x \parallel_{l_2} \leq \frac{c}{\sqrt{k}} \parallel x - x_k \parallel_{l_1}$$

Later on, it will be shown that if Φ is a random matrix with iid Gausssian or Bernoulli random variables as it's entries, then with very high probability, Φ satisfies RIP.

EE5585 Data Compression April 11, 2013

Lecture 21

Instructor: Arya Mazumdar *Scribe: Zhongyu He*

Compressed Sensing

From last lecture, compressed sensing can be boiled down to a simple mathematical problem as the following equation:

$$\Phi x = y$$

where Φ is an m $\times$ N matrix, x is an $N \times 1$ vector, y is an $m \times 1$ vector. And the linear system is undetermined.

Given Φ and y, we need to solve for x ,which is approximately sparse, i.e., x has at most k prominent coordinates. Therefore, we want to design such a matrix Φ that can give us stable solutions.

Stable recovery

Suppose $\hat{x}$ is our estimate. x_k is a vector with k largest coefficient of x.

Suppose $x = \begin{bmatrix} 123 \\ 6.4 \\ -2.5 \\ 193 \\ 4.5 \end{bmatrix}$, $x_2 = \begin{bmatrix} 123 \\ 0 \\ 0 \\ 193 \\ 0 \end{bmatrix}$, then $x - x_k = \begin{bmatrix} 0 \\ 0.4 \\ -2.5 \\ 0 \\ 4.5 \end{bmatrix}$

The stable solution must satisfy the following:

$$\|\hat{x} - x\|_{\ell_2} \leq C\|x - x_k\|_{\ell_1}$$

where C is some constant.

If x has exactly k non-zero elements, then the equation above will give us the exact solution. If not, the error will be bounded by some small value. That's what is called stable recovery.

Basis Pursuit Algorithm(Φ,y)

Now we have an algorithm called Basis Pursuit, with Φ and y as input. It states as following:

$$\min \quad \|z\|_{\ell_1}$$

$$Subject\ to: \Phi z = y$$

This is an optimization problem and can be solved by a linear program.

k-RIP

For any k-sparse vector z, if

$$(1-\delta_k)\|z\|^2_{\ell_2} \leq \|\Phi z\|^2_{\ell_2} \leq (1+\delta_k)\|z\|^2_{\ell_2}$$

is satisfied with the smallest δ_k. Then Φ will be called (k,δ_k)-RIP. RIP is short for Restricted Isometry Property.

Theorem 1(by Candes, Tao 2005/2006)

If Φ is (2k,δ_{2k})-RIP with

$$\delta_{2k} < \sqrt{2} - 1 \approx 0.414$$

that is

$$(1 - 0.414)\|z\|_{\ell_2}^2 \leq \|\Phi z\|_{\ell_2}^2 \leq (1 + 0.414)\|z\|_{\ell_2}^2 \quad \forall\ 2k\texttt{-sparse}\ z$$

Then, basis pursuit solution $\hat{x}$ will satisfy

$$\|\hat{x} - x\|_{\ell_2} \leq (c/\sqrt{k})\|x - x_k\|_{\ell_1}$$

This property guarantees that stable recovery would happen.

Theorem 2

If $m = c_1 k \log N$ then there exists m×N matrix Φ that has $\delta_{2k} < \sqrt{2} - 1$.

If we choose an m×N independent zero-mean random Gaussian matrix satisfying the theorem above, then that matrix with a very high probability will have the RIP. This tells us how to construct Φ, but this is not our concern here.

Proof of **Theorem 1**

Assume $\hat{x} = x + h \Rightarrow h = \hat{x} - x$, where h is the error vector.

We will bound from above $\|h\|_{\ell_2}$

First of all,

$$\|x\|_{\ell_1} \geq \|\hat{x}\|_{\ell_1} = \|x + h\|_{\ell_1}$$

Now assume a vector v and a subset T $\subseteq$ {1,2,$\cdots$ N}. v_T is the projection of v on T.

For example:

$$v = \begin{bmatrix} 5 \\ 3 \\ 4 \\ -1 \\ 2 \end{bmatrix}, \text{T} = \{1,4,5\}, \text{then } v_t = \begin{bmatrix} 5 \\ 6 \\ 0 \\ -1 \\ 2 \end{bmatrix}$$

$$\begin{aligned}
\|x\|_{\ell_1} &\geq \|x_{T_0} + h_{T_0} + x_{T_0^c} + h_{T_0^c}\|_{\ell_1} && (1)\\
&= \|x_{T_0} + h_{T_0}\|_{\ell_1} + \|x_{T_0^c} + h_{T_0^c}\|_{\ell_1} && (2)\\
&\geq \|x_{T_0}\|_{\ell_1} - \|h_{T_0}\|_{\ell_1} - \|x_{T_0^c}\|_{\ell_1} + \|h_{T_0^c}\|_{\ell_1} && (3)\\
\Rightarrow \|x_{T_0}\|_{\ell_1} + \|x_{T_0^c}\|_{\ell_1} &\geq \|x_{T_0}\|_{\ell_1} - \|h_{T_0}\|_{\ell_1} - \|x_{T_0^c}\|_{\ell_1} + \|h_{T_0^c}\|_{\ell_1} && (4)\\
\Rightarrow \|h_{T_0^c}\|_{\ell_1} &\leq \|h_{T_0}\|_{\ell_1} + 2\|x_{T_0^c}\|_{\ell_1} && (5)
\end{aligned}$$

where T_0 is the k-largest coordinate of x.

*Reasonnings for (2)(4)

For example,

$$\hat{x}= \begin{bmatrix} 9 \\ 4 \\ 2 \\ 3 \\ 1 \\ -1 \\ -9 \\ 6 \end{bmatrix}, \hat{x}_{T_0}= \begin{bmatrix} 9 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -9 \\ 6 \end{bmatrix}, \hat{x}_{T_0^c}= \begin{bmatrix} 0 \\ 4 \\ 2 \\ 3 \\ 1 \\ -1 \\ 0 \\ 0 \end{bmatrix},$$

$$\begin{aligned} \|\hat{x}\|_{\ell_1} &= |9| + |4| + |2| + |3| + |1| + |-1| + |-9| + |6| \\ &= \|\hat{x}_{T_0}\|_{\ell_1} + \|\hat{x}_{T_0^c}\|_{\ell_1} \end{aligned}$$

****Resonings for (3)**

By Triangle Inequality

Lemma

If z, z' are two vectors that are k_1-sparse and k_2-sparse respectively; moreover, the coordinates where z, z' are non-zero do not overlap. Then

$$| < \Phi z, \Phi z' > | \leq \delta_{k_1+k_2} \|z\|_{\ell_2} \|z'\|_{\ell_2}$$

Proof

$$\begin{aligned} | < \Phi z, \Phi z' > | &= \frac{1}{4} \left[\|\Phi z + \Phi z'\|_{\ell_2}^2 - \|\Phi z - \Phi z'\|_{\ell_2}^2 \right] \\ &= \frac{1}{4} \left[\|\Phi(z + z')\|_{\ell_2}^2 - \|\Phi(z - z')\|_{\ell_2}^2 \right] \end{aligned}$$

$$(1 - \delta_{k_1+k_2}) \|z \pm z'\|_{\ell_2}^2 \leq \|\Phi(z \pm z')\|_{\ell_2}^2 \leq (1 + \delta_{k_1+k_2}) \|z \pm z'\|_{\ell_2}^2$$

$$\begin{aligned} \Rightarrow | < \Phi z, \Phi z' > | &\leq \frac{1}{4} \left[(1 + \delta_{k_1+k_2}) \|z + z'\|_{\ell_2}^2 - (1 - \delta_{k_1+k_2}) \|z + z'\|_{\ell_2}^2 \right] \\ &= \frac{1}{2} \delta_{k_1+k_2} \|z + z'\|_{\ell_2}^2 \end{aligned}$$

Now

$$\begin{aligned} \left| \frac{< \Phi z, \Phi z' >}{\|z\|_{\ell_2} \|z'\|_{\ell_2}} \right| &= | < \Phi \frac{z}{\|z\|_{\ell_2}}, \Phi \frac{z'}{\|z'\|_{\ell_2}} > | \\ &\leq \frac{1}{2} \delta_{k_1+k_2} \left\| \frac{z}{\|z\|_{\ell_2}} + \frac{z'}{\|z'\|_{\ell_2}} \right\|_{\ell_2}^2 \\ &= \delta_{k_1+k_2} \\ \Rightarrow | < \Phi z, \Phi z' > | &\leq \delta_{k_1+k_2} \|z\|_{\ell_2} \|z'\|_{\ell_2} \end{aligned}$$

Lemma proved. ■

Let's come back to the proof of **Theorem 1**.
T_0 is the k-largest coefficients of x.
T_1 is the k-largest in absolute value coefficients of $h_{T_0^c}$.
T_2 is the next k-largest in absolute value coefficients of $h_{T_0^c}$.

$$\vdots$$

Next, we will show that both $\|h_{T_0\cup T_1}\|$ and $\|h_{(T_0\cup T_1)^c}\|$ are bounded.
Note that for any $j \geq 2$,

$$\begin{aligned}\|h_{T_j}\|_{\ell_2} &\leq \sqrt{k}\ \texttt{max value in}\ h_{T_j}\ \texttt{(Definition of l2 norm)}\\ &\leq \sqrt{k}\ \texttt{average absolute value in}\ h_{T_{j-1}}\\ &\leq \frac{\sqrt{k}}{k}\|h_{T_{j-1}}\|_{\ell_1}\end{aligned}$$

Now

$$\begin{aligned}\|h_{(T_0\cup T_1)^c}\|_{\ell_2} &= \|\sum_{j\geq 2} h_{T_j}\|_{\ell_2}\\ &\leq \sum_{j\geq 2}\|h_{T_j}\|_{\ell_2}\\ &= \frac{1}{\sqrt{k}}\sum_{j\geq 1}\|h_{T_j}\|_{\ell_1}\\ &= \frac{1}{\sqrt{k}}\|h_{T_0^c}\|_{\ell_1}\\ &\leq \frac{1}{\sqrt{k}}(\|h_{T_0}\|_{\ell_1} + 2\|x_{T_0^c}\|_{\ell_1}) \qquad (6)\end{aligned}$$

Because

$$\begin{aligned}\|z\|_{\ell_1} &\geq \|z\|_{\ell_2}\\ |z_1| + |z_2| &\geq \sqrt{|z_1|^2 + |z_2|^2}\\ \|z\|_{\ell_2} &\geq \frac{\|z\|_{\ell_1}}{\sqrt{\texttt{number of elements}}}\\ \Rightarrow \sqrt{z_1^2 + z_2^2 + \cdots + z_k^2} &\geq \frac{|z_1| + |z_2| + \cdots + |z_k|}{\sqrt{k}}\end{aligned}$$

So from(6),

$$\begin{aligned}\|h_{(T_0\cup T_1)^c}\|_{\ell_2} &\leq \|h_{T_0}\|_{\ell_2} + \frac{2}{\sqrt{k}}\|x_{T_0^c}\|_{\ell_1}\\ \Rightarrow \|h_{(T_0\cup T_1)^c}\|_{\ell_2} &\leq \|h_{(T_0\cup T_1)}\|_{\ell_2} + \frac{2}{\sqrt{k}}\|x_{T_0^c}\|_{\ell_1} \qquad (7)\end{aligned}$$

Now we have

$$\|h_{(T_0\cup T_1)}\|_{\ell_2}^2 \leq \frac{1}{1-\delta_{2k}}\|\Phi h_{(T_0\cup T_1)}\|_{\ell_2}^2 \qquad (8)$$

$$\begin{aligned}\|\Phi h_{(T_0\cup T_1)}\|_{\ell_2}^2 &=< \Phi h_{(T_0\cup T_1)}, \Phi h_{(T_0\cup T_1)} >\\ &=< \Phi h_{(T_0\cup T_1)}, \Phi(h - h_{(T_0\cup T_1)^c}) >\end{aligned}$$

Since $\Phi h = \Phi(\hat{x} - x) = \Phi\hat{x} - \Phi x = y - y = 0$

$$\begin{aligned}
\Rightarrow \|\Phi h_{(T_0 \cup T_1)}\|_{\ell_2}^2 &=< \Phi h_{(T_0 \cup T_1)}, -\Phi h_{(T_0 \cup T_1)^c} > \\
&=< \Phi h_{(T_0 \cup T_1)}, -\sum_{j \geq 2} \Phi h_{T_j} > \\
&\leq | < \Phi h_{T_0}, -\sum_{j \geq 2} \Phi h_{T_j} > | + | < \Phi h_{T_1}, -\sum_{j \geq 2} \Phi h_{T_j} > | \\
&\leq \sum_{j \geq 2} \left[| < \Phi h_{T_0}, \Phi h_{T_j} > | + | < \Phi h_{T_1}, \Phi h_{T_j} > | \right] \\
&\leq \sum_{j \geq 2} (\delta_{2k} \|h_{T_0}\|_{\ell_2} \|h_{T_1}\|_{\ell_2} + \delta_{2k} \|h_{T_1}\|_{\ell_2} \|h_{T_j}\|_{\ell_2})
\end{aligned}$$

Since $\sqrt{a} + \sqrt{b} \leq \sqrt{2(a+b)}$

$$\begin{aligned}
\Rightarrow \|\Phi h_{(T_0 \cup T_1)}\|_{\ell_2}^2 &\leq \delta_{2k} 2 \|h_{(T_0 \cup T_1)}\|_{\ell_2} (\|h_{(T_0 \cup T_1)}\|_{\ell_2} + \frac{2}{\sqrt{k}} \|x_{T_0^c}\|_{\ell_1}) \\
\texttt{from (8)} \quad \Rightarrow \|h_{(T_0 \cup T_1)}\|_{\ell_2}^2 &\leq \frac{1}{1 - \delta_{2k}} \left[\delta_{2k} 2 \|h_{(T_0 \cup T_1)}\|_{\ell_2} (\|h_{(T_0 \cup T_1)}\|_{\ell_2} + \frac{2}{\sqrt{k}} \|x_{T_0^c}\|_{\ell_1}) \right] \\
\|h_{(T_0 \cup T_1)}\|_{\ell_2} (1 - \frac{\sqrt{2}\delta_{2k}}{1 - \delta_{2k}}) &\leq \frac{2}{\sqrt{k}} \|x_{T_0^c}\|_{\ell_1}) \\
\|h_{(T_0 \cup T_1)}\|_{\ell_2} &\leq \frac{1}{(1 - \frac{\sqrt{2}\delta_{2k}}{1 - \delta_{2k}})} \frac{2}{\sqrt{k}} \|x_{T_0^c}\|_{\ell_1}) \qquad (9)
\end{aligned}$$

$$\begin{aligned}
\|h\|_{\ell_2} &\leq \|h_{(T_0 \cup T_1)}\|_{\ell_2} + \|h_{(T_0 \cup T_1)^c}\|_{\ell_2} \\
&= 2\|h_{(T_0 \cup T_1)^c}\|_{\ell_2} + \frac{2}{\sqrt{k}} \|x_{T_0^c}\|_{\ell_1} \\
\texttt{from (9)} \quad &\Rightarrow \leq (\frac{2}{(1 - \frac{\sqrt{2}\delta_{2k}}{1 - \delta_{2k}})} + 1) \frac{2}{\sqrt{k}} \|x_{T_0^c}\|_{\ell_1} \\
\Rightarrow \|\hat{x} - x\|_{\ell_2} &\leq (\frac{2}{(1 - \frac{\sqrt{2}\delta_{2k}}{1 - \delta_{2k}})} + 1) \frac{2}{\sqrt{k}} \|x - x_2\|_{\ell_1}
\end{aligned}$$

where $\left(\frac{2}{(1 - \frac{\sqrt{2}\delta_{2k}}{1 - \delta_{2k}})} + 1 \right)$ is positive for any $\delta_{2k} > \sqrt{2} - 1$

QED (End of the Proof for **Theorem 1**)

■

EE5585 Data Compression April 16, 2013

Lecture 22

Instructor: Arya Mazumdar *Scribe: Cheng-Yu Hung*

Review of Compressed Sensing

Consider a general linear measurement process that produces an $m \times 1$ observation vector $\mathbf{y}$ in terms of a $N \times 1$ signal vector $\mathbf{x}$ and the columns of a $m \times N (m \ll N)$ measurement matrix Φ. It can be expressed as

$$\Phi\mathbf{x} = \mathbf{y} \tag{1}$$

It is noted that the signal vector $\mathbf{x}$ is an approximately k-sparse vector which has at most $k(\ll N)$ non-zero entries.

Basic Pursuit

Given $\mathbf{y}$ and Φ, recovery of the unknown signal vector $\mathbf{x}$ can be pursued by finding the sparsest estimate of $\mathbf{x}$ which has the constraint $\Phi\mathbf{x} = \mathbf{y}$, i.e.,

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & ||\mathbf{x}||_{l_0} \\ \text{subject to} \quad & \Phi\mathbf{x} = \mathbf{y}. \end{aligned}$$

However, this is an NP-hard problem. Convex relaxation methods cope with the intractability of the above formulation by approximating the l_0 norm by the convex l_1 norm. This is a well-known algorithm which is called basic pursuit. The above formulation is changed to

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & ||\mathbf{x}||_{l_1} \\ \text{subject to} \quad & \Phi\mathbf{x} = \mathbf{y}. \end{aligned} \tag{2}$$

Many linear programmings can solve this problem, such as the simplex method or interior point methods.

Stable Recovery and Restricted Isometry Property (RIP)

The RIP imposes that there exists a $0 < \delta_{2k} < 1$ such that for any $\mathbf{z}$ that has at most $2k$ nonzero entries,

$$(1 - \delta_{2k})||\mathbf{z}||_{l_2}^2 \leq ||\Phi\mathbf{z}||_{l_2}^2 \leq (1 + \delta_{2k})||\mathbf{z}||_{l_2}^2 \tag{3}$$

Candes et al. (2006) have shown that if Φ satisfies RIP with $\delta_{2k} \leq \sqrt{2} - 1$, then the solution $\hat{\mathbf{x}}$ to equation (2) will achieve the stable recovery,

$$||\hat{\mathbf{x}} - \mathbf{x}||_{l_2}^2 \leq \frac{c}{\sqrt{k}}||\hat{\mathbf{x}} - \mathbf{x}_k||_{l_1}^2, \tag{4}$$

where $\mathbf{x}_k$ is the restriction of $\mathbf{x}$ to its k largest entries.

Design Good Measurement Matrices Φ

One of main issues is to find a matrix Φ with RIP: $\delta_{2k} < \sqrt{2} - 1$. Given the $N \times 1$ vector $\mathbf{x}$ and $m \times N (m \ll N)$ measurement matrix Φ, we denote a $m \times 2K$ matrix Φ_I, and a $2k \times 1$ vector $\mathbf{z} = \mathbf{x}_I$, where $I \subseteq \{1, \cdots, N\}$, $|I| = 2k$. Since $\delta_{2k} < \sqrt{2} - 1$, the RIP becomes as the following:

$$0.586 = (1 - \delta_{2k}) \leq \frac{||\Phi\mathbf{z}||_{l_2}^2}{||\mathbf{z}||_{l_2}^2} \leq (1 + \delta_{2k}) = 1.414. \tag{5}$$

We want any $2k$ columns of Φ satisfying the above inequalities (5). Actually, (5) is also equivalent to

$$0.586 \leq \text{eigenvalues of } \Phi_I^T \Phi_I \leq 1.414. \tag{6}$$

or

$$0.586 \leq \text{sigular values of } \Phi_I \leq 1.414. \tag{7}$$

But the question is what kind of matrix Φ has this property of (7)?
The answer is that the matrix Φ can be a random matrix chosen in the following way:

$$\Phi = \frac{1}{\sqrt{m}} \begin{bmatrix} \phi_{11} & \cdots & \phi_{1N} \\ \vdots & \ddots & \vdots \\ \phi_{m1} & \cdots & \phi_{mN} \end{bmatrix}, \text{where } \phi_{ij} \overset{iid}{\sim} \mathcal{N}(0,1). \tag{8}$$

with high probability for any I such that Φ_I satisfies

$$0.586 \leq \text{eigenvalues of } \Phi_I^T \Phi_I \leq 1.414.$$

In particular the high probability is $1 - e^{-\alpha m}$, for some $\alpha > 0$.
Then,

$$\begin{aligned}
&\Pr(\text{RIP-2k isn't satisfied}) && (9)\\
&= \Pr(\exists \text{ a set of 2k columns for which (7) is not satisfied}) && (10)\\
&\leq \binom{N}{2k} e^{-\alpha m} && (11)\\
&= (\frac{Ne}{2k})^{2k} e^{-\alpha m} && (12)\\
&= e^{2k \log(\frac{Ne}{2k}) - \alpha m} && (13)\\
&\text{if } \alpha m \geq 2 \cdot 2k \log(\frac{Ne}{2k}) && (14)\\
&\leq e^{\frac{-\alpha m}{2}} \to 0 \text{ as } m \to \infty && (15)
\end{aligned}$$

In summary, suppose that the entries of the $m \times 2k$ matrix Φ_I are i.i.d. Gaussian with zero mean and variance $\frac{1}{m}$. Then, with high probability, Φ satisfies the required RIP condition for stable recovery to hold, provided that

$$m > \frac{4}{\alpha} k \log(\frac{Ne}{2k}). \tag{16}$$

Moreover, sensing matrices whose entries are iid from a Bernoulli distribution ($+\frac{1}{\sqrt{m}}$ with prob. 1/2 and $-\frac{1}{\sqrt{m}}$ with prob. 1/2), columns normalized to a unit norm, also obey the RIP given equation (16). It is noted that since $m > \frac{4}{\alpha} k \log(\frac{Ne}{2k})$, the observation signal's dimension is $m = \mathcal{O}(k \log \frac{N}{k})$, while the dimension of input signal is N.

Differential Encoding

In many sources we are interested in, the sampled source output $\{x_n\}$ does not change a lot from one sample to the next. This means that the variance of the sequence of difference $\{d_n = x_n - x_{n-1}\}$ are

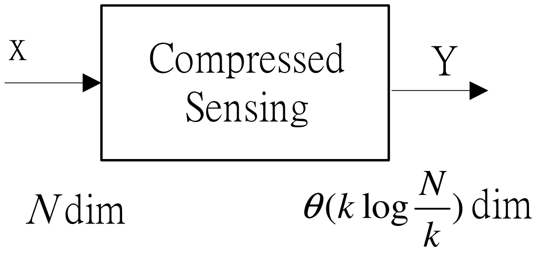


Figure 1: Compressed sensing

substantially smaller than that of the source output sequence. In other words, for the correlated data the distribution of d_n is highly peaked at zero. It is useful to encode the difference from the sample to the next rather than encoding the actual sample values since the variance of quantization error in the differences is less than in the actula output samples. This technque is called *$differential$ encoding*.

Quantize with Differential Encoding

Consider a sequence $\{x_n\}$ in Figure 2. A difference sequence $\{d_n\}$ is generated by taking the difference $x_n - x_{n-1}$. The difference sequence is quantized to obtain the sequence $\{\hat{d}_n\}$. So,

$$\begin{aligned} d_0 &= x_0 && (17)\\ d_1 &= x_1 - x_0 && (18)\\ d_2 &= x_2 - x_1 && (19)\\ &\vdots && (20)\\ d_n &= x_n - x_{n-1}. && (21) \end{aligned}$$

The above equations can be written as a matrix form, i.e.,

$$\begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & & \ddots & \\ & & & -1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ \\ \vdots \\ \\ x_n \end{bmatrix} = \begin{bmatrix} d_0 \\ \\ \vdots \\ \\ d_n \end{bmatrix} \text{ or } \begin{bmatrix} 1 & & & & \\ 1 & 1 & & & \\ 1 & 1 & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \begin{bmatrix} d_0 \\ \\ \vdots \\ \\ d_n \end{bmatrix} = \begin{bmatrix} x_0 \\ \\ \vdots \\ \\ x_n \end{bmatrix} \tag{22}$$

And

$$\hat{d}_n = Q[d_n] = d_n + q_n, \text{ where } q_n \text{ is the quantization error.} \tag{23}$$

At the receiver, the reconstructed sequence $\{\hat{x}_n\}$ is obtained by adding $\hat{d}_n$ to the previous reconstructed value $\hat{x}_{n-1}$:

$$\hat{x}_n = \hat{x}_{n-1} + \hat{d}_n. \tag{24}$$

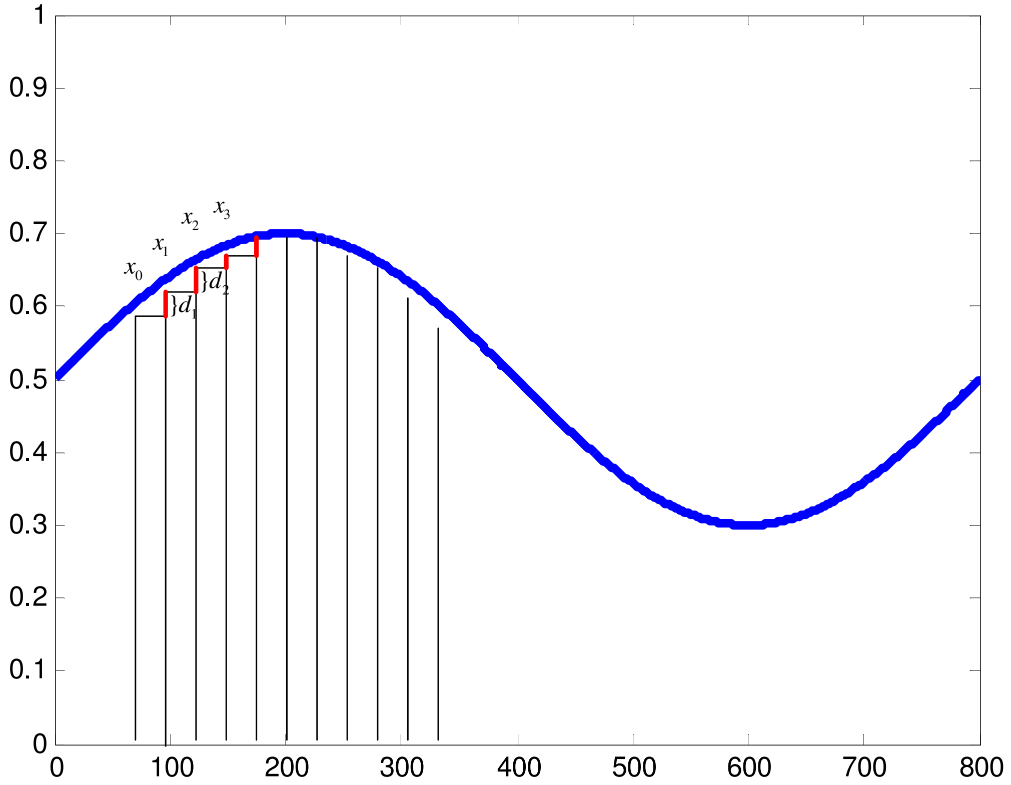


Figure 2: Output samples $\{x_n\}$ and difference samples $\{d_n\}$

Let us assume that both transmitter ans receiver start with the same value x_0, that is, $\hat{x}_0 = x_0$. Then, follow the quantization and reconstruction process:

$$\hat{x}_0 = d_0 + q_0 \tag{25}$$

$$\hat{x}_1 = \hat{x}_0 + \hat{d}_1 = d_0 + q_0 + d_1 + q_1 = x_1 + q_0 + q_1 \tag{26}$$

$$\hat{x}_2 = \hat{x}_1 + \hat{d}_2 = d_0 + d_1 + q_0 + q_1 + d_2 + q_2 = x_2 + q_0 + q_1 + q_2 \tag{27}$$

$$\vdots \tag{28}$$

$$\hat{x}_n = d_0 + d_1 + \cdots + d_n + q_0 + q_1 + \cdots + q_n = x_n + \sum_{i=0}^{n} q_i \tag{29}$$

So, at the ith iteration we get

$$\hat{x}_i = x_i + \sum_{j=0}^{i} q_j. \tag{30}$$

We can see that the quantization error accumulates as the process continues. According to central limit

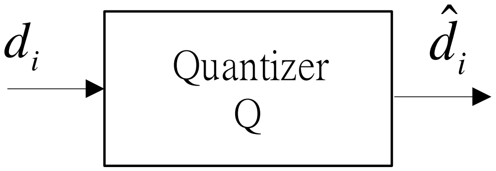


Figure 3: Quantizer of differential encoding

theorem, the sum of these quantization error is Gaussian noise and its mean is zero, but in fact, before that happens, the finite precision of machines causes the reconstructed value to overflow.
Therefore, instead of $d_n = x_n - x_{n-1}$, we use $d_n = x_n - \hat{x}_{n-1}$. By use of this new differencing operation, we repeat the quantization and reconstruction process. Assume that $\hat{x}_0 = x_0$,

$$\begin{aligned} d_1 &= x_1 - x_0 && (31) \\ \hat{d}_1 &= d_1 + q_1 && (32) \\ \hat{x}_1 &= x_0 + \hat{d}_1 = x_0 + d_1 + q_1 = x_1 + q_1 && (33) \\ d_2 &= x_2 - \hat{x}_1 && (34) \\ \hat{d}_2 &= d_2 + q_2 && (35) \\ \hat{x}_2 &= \hat{x}_1 + \hat{d}_2 = \hat{x}_1 + d_2 + q_2 = x_2 + q_2 && (36) \end{aligned}$$

So, at the ith iteration, we have

$$\hat{x}_i = x_i + q_i, \tag{37}$$

and there is no accumulation of the quantization error. The quantization error q_i is the quantization noise incurred by the quantization of the ith difference and it is significantly less than the quantization error for the original sequence. Thus, this procedure leads to an overall reduction of the quantization error and then we can use fewer bits with a differential encoding to attain the same distortion.

Subband Coding

Consider the sequence $\{x_n\}$ in the Figure 4. We see that while there is a significant amount of sample-to-sample variations, there's also an underlying long-term trend shown by the blue line that varied slowly. One way to extract this trend is to use moving window to average the sample value. Let us use a window of size two and generate a new sequence $\{y_n\}$ by averaging neighboring values of x_n:

$$y_n = \frac{x_n + x_{n-1}}{2}. \tag{38}$$

The consecutive values of y_n will be closer to each other than the consecutive values of x_n. Thus, the $\{y_n\}$ can be coded more efficiently using differential encoding than the sequence $\{x_n\}$. However, we want to encode the sequence $\{x_n\}$, not $\{y_n\}$. Therefire, we need another sequence $\{z_n\}$:

$$z_n = x_n - y_n = x_n - \frac{x_n + x_{n-1}}{2} = \frac{x_n - x_{n-1}}{2}. \tag{39}$$

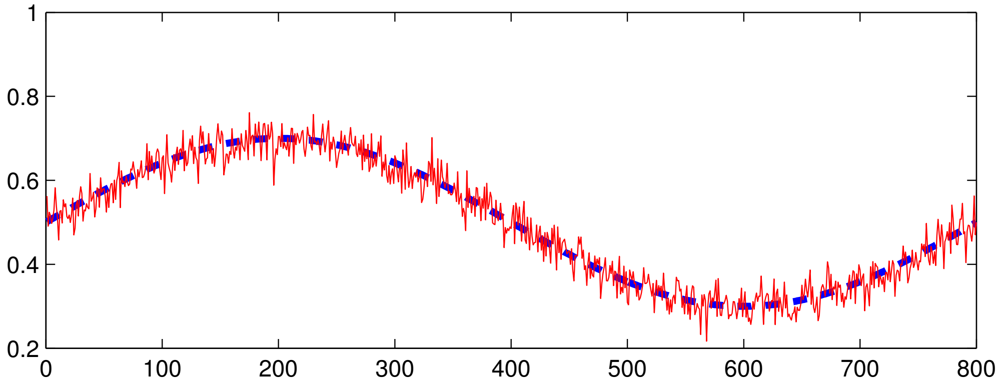


Figure 4: A rapidly changong source output that contains a long-run component with slow variations.

Then, the sequences $\{y_n\}$ and $\{z_n\}$ can be coded independently of each other.
Notice that we use the same number of bits for each value of y_n and z_n, but the number of elements in each of the sequences $\{y_n\}$ and $\{z_n\}$ is the same as the number of elements in the original sequence $\{x_n\}$. Although we are using the same number of bits of bits per sample, we are transmitting twice as many samples ans doubling the bit rate. We can avoid this by sending every other value of y_n and z_n. Let's divide the sequences $\{y_n\}$ into $\{y_{2n}\}$ and $\{y_{2n-1}\}$, and similarly, divide $\{z_n\}$ into $\{z_{2n}\}$ and $\{z_{2n-1}\}$. If we transmit either the even-numbered subsequencse or odd-numbered subsequences, we would transmit only as many elements as in the original sequence.

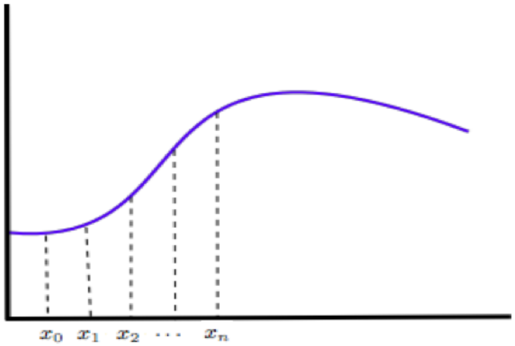
EE5585 Data Compression April 18, 2013

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

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

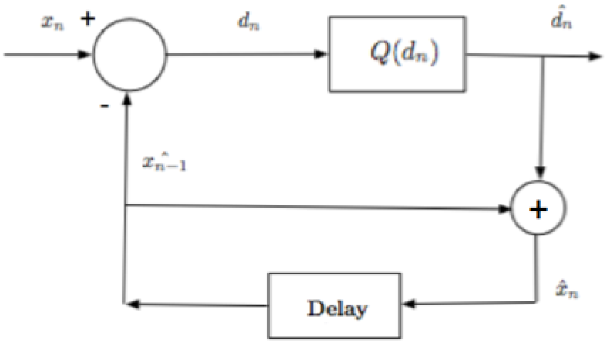
If you make an error in the d_n the noise in x_n will be accumulated as follows

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

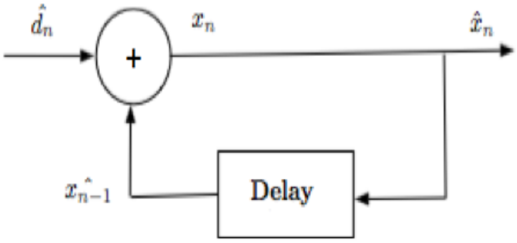
The quantization noise q_n can be positive or negative and in the long term we expect it to add up to zero, but what happens with high probability is that the range of the error becomes too great for us to handle. Therefore, we adopt the following strategy

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

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

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

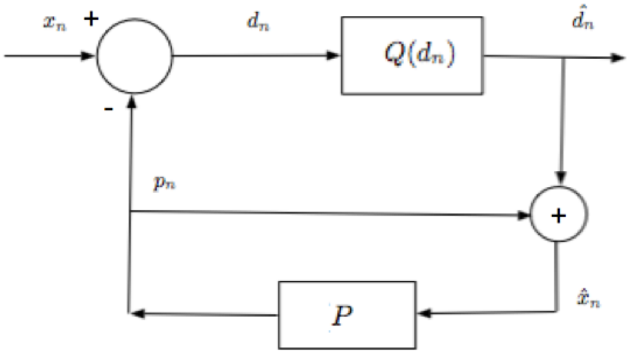
We see that in general

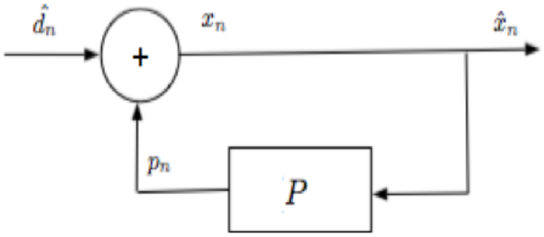
$$\begin{aligned}\hat{x_n} &= \hat{d_n} + \hat{x}_{n-1}\\ &= d_n + q_n + \hat{x}_{n-1}\\ &= x_n + q_n\end{aligned}$$

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

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

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

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

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

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

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

Now the variance of d_n becomes

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

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

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

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

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

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

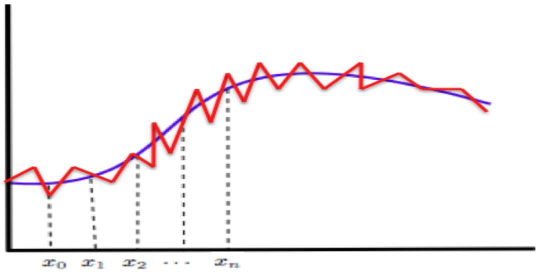
$$\begin{pmatrix} R_{xx}(0) & R_{xx}(1) & \dots & R_{xx}(N-1) \\ R_{xx}(1) & R_{xx}(0) & & \vdots \\ \vdots & & \ddots & \vdots \\ R_{xx}(N-1) & & \dots & R_{xx}(0) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} R_{xx}(1) \\ R_{xx}(2) \\ \vdots \\ R_{xx}(N)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

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

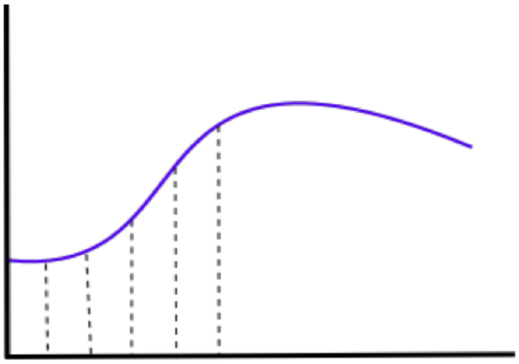
$$Ra = P$$
$$a = R^{-1}P$$

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.

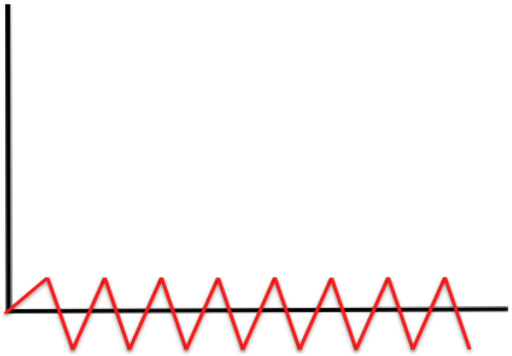
Now suppose that our signal is rapidly changing as shown below



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).

$$\begin{aligned} y_n &= \frac{x_n+x_{n-1}}{2} \\ z_n &= \frac{x_n-x_{n-1}}{2} \end{aligned}$$

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

$$\begin{aligned} y_{2n} &= \frac{x_{2n}+x_{2n-1}}{2} \\ z_{2n} &= \frac{x_{2n}-x_{2n-1}}{2} \end{aligned}$$

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

$$\begin{aligned} y_{2n} + z_{2n} &= x_{2n} \\ y_{2n} - z_{2n} &= x_{2n-1} \end{aligned}$$

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

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

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.

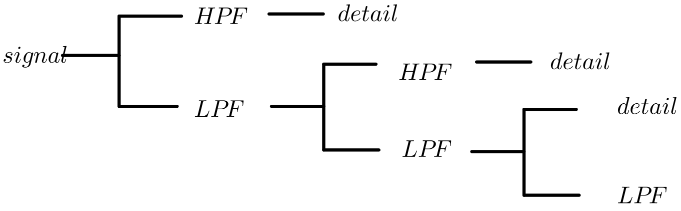
EE5585 Data Compression April 23, 2013

Lecture 24

Instructor: Arya Mazumdar *Scribe: Yuanhang Wang*

Recap - Sub-band Coding

In the last class we talked about sub-band coding. In the first step of it, signal goes into two parts, high frequency component (details, local property) and low frequency component (average values, global property). Then after several LPFs we will end with the average value of this signal. The idea is that we don't need some of the details that comes from HPFs, so we throw them out and do the data compression.



One possible way to achieve this is doing the DFT in each step. The main problem of DFT is that we may not have a single matrix to achieve the whole steps. Suppose we have some changes in time domain. Then we cannot know how it changes from observing frequency domain. According to *uncertainty principle* in Fourier analysis, a function f and its Fourier transform $\mathcal{F}$ cannot be simultaneously well localized. To take care of that, we need something else called *wavelets*.

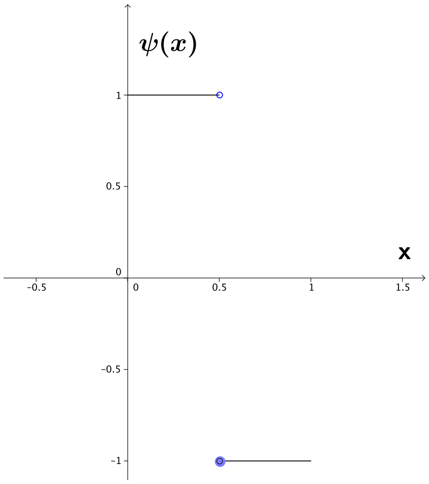
Wavelets

Wavelet is also one of data transform coding techniques. It is a kind of multi-resolution coding.

Let's look at the Haar Wavelet first.

Definition 1 (Mother function of Haar wavelet $\psi(x)$)

$$\psi(x) = \begin{cases} 1, & 0 \leq x < \frac{1}{2} \\ \\ -1, & \frac{1}{2} \leq x < 1 \\ \\ 0, & otherwise \end{cases}$$



Now our idea is to find some basis space for any function f.

Here let's consider the space of all square-integrable real-valued functions H. A function is said to be *square-integrable* if

$$\int_{-\infty}^{+\infty} |f(x)|^2 dx < \infty \tag{1}$$

Definition 2 (Hilbert Space) *A Hilbert space H is an inner product space that is also a complete metric space with respect to the distance function induced by the inner product.*

In space H, two square-integrable real-valued functions f and g have an *inner product*, which is defined as

$$< f, g >= \int f(x)g(x)dx \tag{2}$$

Norm is defined as $\|f\|$, where

$$\|f\|^2 = \int |f(x)|^2 dx =< f, f > \tag{3}$$

Our idea is to change the basis of $f(x)$ and represent it in other basis, say, $g(x)$. Then the expression will be:

$$f(x) = \sum_k C_k g_k(x), k = 0, 1, 2, ...$$

Also we have the concept of *Lebesgue Space*: The space of all measurable functions f on the interval [T_1, T_2] satisfying

$$\int_{T_1}^{T_2} |f(x)|^2 dx < \infty$$

is called the L^2 space, $L^2[T_1,T_2]$.

Let's define

$$\psi_{0,0}(x) = \psi(x)$$

and

$$\psi_{j,k}(x) = 2^{\frac{j}{2}} \psi(2^j x - k), where\, j \in \mathbb{N}, k \in \mathbb{Z}.$$

It's clear that

$$\psi_{j,k}(x) = \begin{cases} 2^{\frac{j}{2}}, & \frac{k}{2^j} \leq x < \frac{k+\frac{1}{2}}{2^j} \\ \\ 2^{\frac{-j}{2}}, & \frac{k+\frac{1}{2}}{2^j} \leq x < \frac{k+1}{2^j} \end{cases}$$

Here the set $\{\psi_{j,k}, j \in \mathbb{N}, k \in \mathbb{Z}\}$ is an *orthonormal basis* in $L^2[\mathbb{R}]$.

Let's verify that this basis is orthonormal. We can notice that:

$$\int_{-\infty}^{+\infty} |\psi_{j,k}(x)|^2 dx = \int_{-\infty}^{+\infty} (2^{\frac{j}{2}})^2 |\psi(2^j x - k)|^2 dx = \int_{\frac{k}{2^j}}^{\frac{k+1}{2^j}} 2^j dx = 2^j \frac{1}{2^j} = 1$$

Also it's easy to prove that:

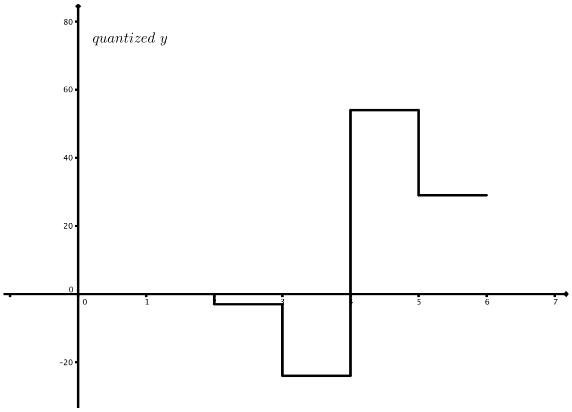
$$\int_{-\infty}^{+\infty} \psi_{j,k}(x)\psi_{j',k'}(x)dx = 0, if\, j \neq j' and\, k \neq k'$$

So $\{\psi_{j,k}, j \in \mathbb{N}, k \in \mathbb{Z}\}$ forms an orthonormal basis.

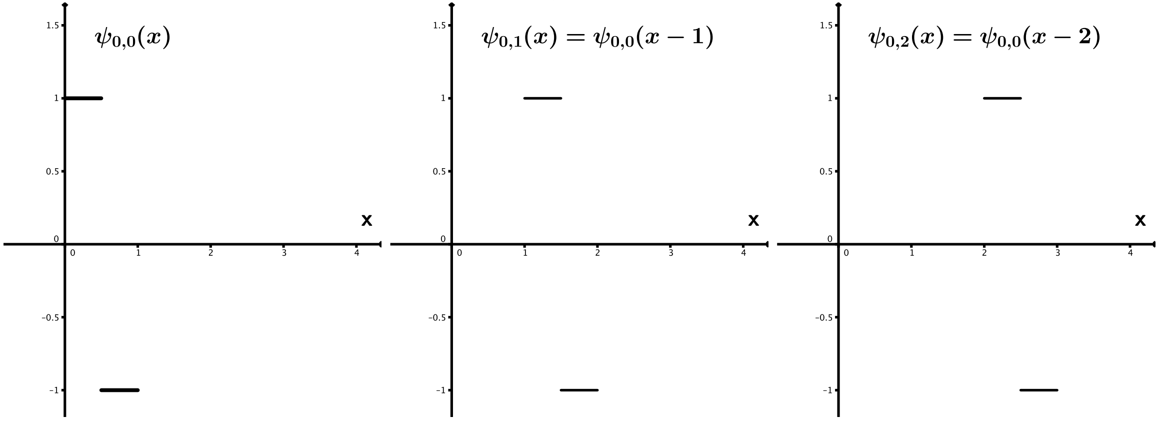
Now let's try to use this basis to represent function $f(x)$.

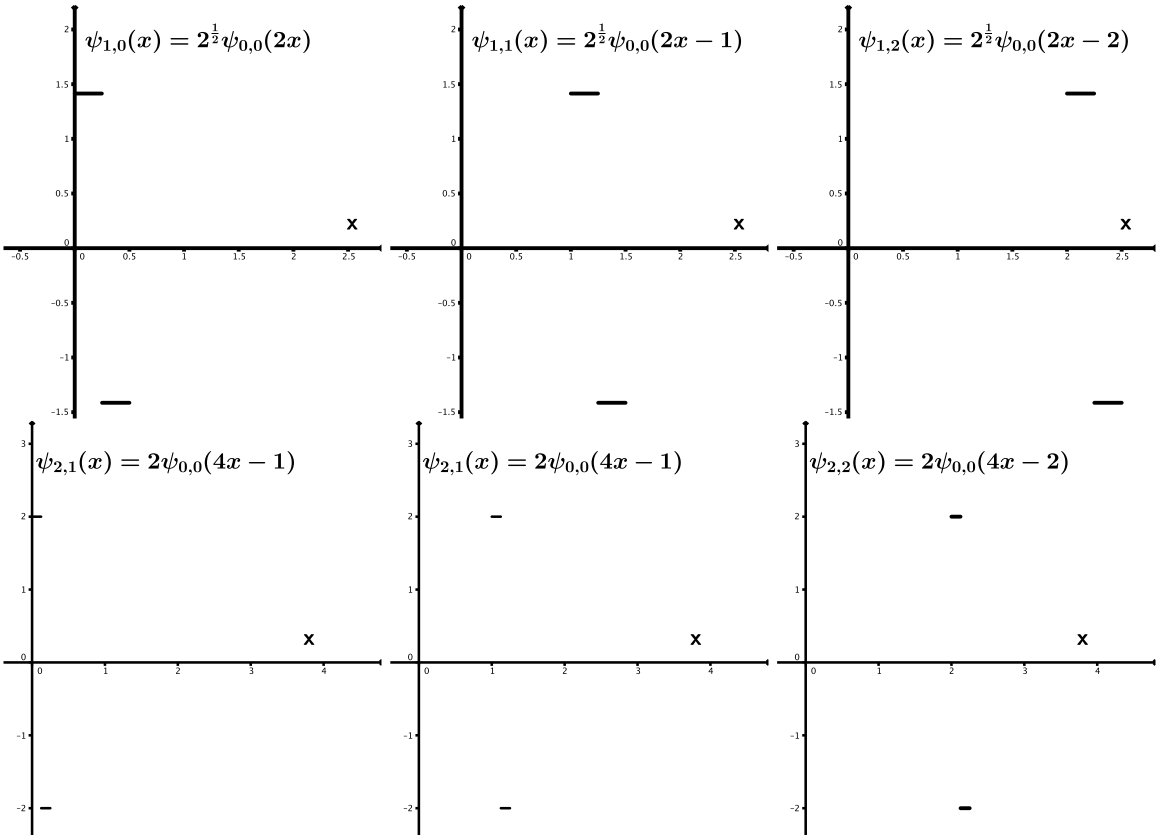
$$f(x) = \sum_{j \geq 0,\, k \in \mathbb{Z}} c_{j,k}\psi_{j,k}(x)$$

In practice, we take discrete points of $f(x)$ and use $\psi(x)$ to represent it. For example, y= [0 -3 -24 54 29 ...], then our graph would be:



Now let's look at the translating and scaling operations on mother function.





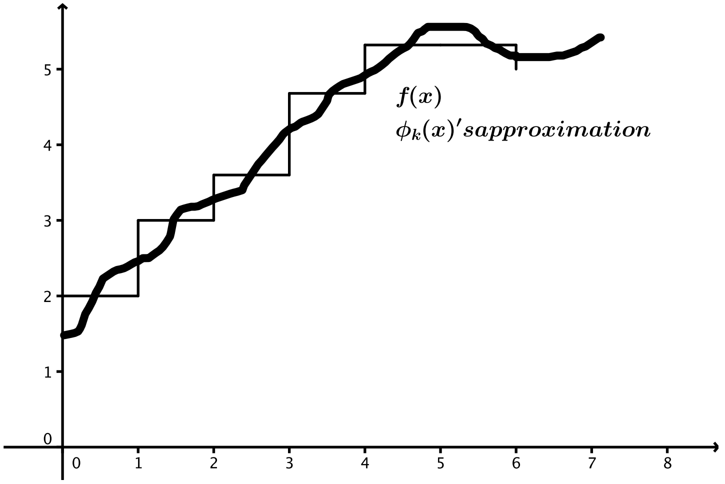
The rows show how translating works, and the columns show how scaling works.

Definition 3 (Scaling function of Haar wavelet $\phi(x)$)

$$\phi(x) = \begin{cases} 1, & 0 \leq x < 1 \\ \\ 0, & otherwise \end{cases}$$

We use $\phi_k(x)$ or $\phi_{0,k}(x)$ to denote $\phi(x-k)$, and $\phi_{1,k}(x) = \phi_{0,k}(2x)$. Here are the steps that we use to approximate an arbitrary function $f(x)$:

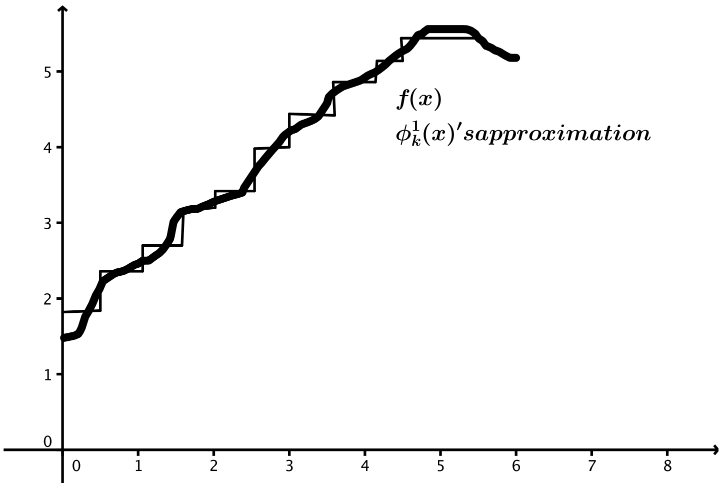
1. Approximate only using $\phi_k(x)$(translating):



The approximation $\phi_f^0(x) = \sum C_{0,k}\phi(x-k)$, where $C_{0,k} = \int f(x)\phi(x-k)dx = \int_k^{k+1} f(x)dx$. $C_{0,k}$ is the average value of $f(x)$ in each interval.

2. Based on step one, we do the scaling:

$$\phi_{1,k}(x) = \phi(2x-k) = \begin{cases} 1, & \frac{k}{2} \le x < \frac{k+1}{2} \\ \\ 0, & otherwise \end{cases}$$

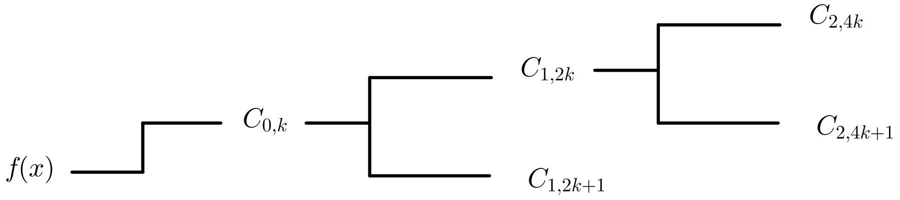


This approximation $\phi_f^1(x) = \sum C_{1,k}\phi_{1,k}(x)$, where $C_{1,k} = 2\int f(x)\phi_{1,k}(x)dx = 2\int_{\frac{k}{2}}^{\frac{k+1}{2}} f(x)dx$. $C_{1,k}$ is still the average value of $f(x)$ in each smaller interval.

We can see that:

$$C_{1,2k} + C_{1,2k+1} = 2\int_k^{k+\frac{1}{2}} f(x)dx + 2\int_{k+\frac{1}{2}}^{k+1} f(x)dx = 2\int_k^{k+1} f(x)dx = C_{0,k}$$

So we may apply this property to construct a kind of sub-band coding like:



This is a simple wavelet called Haar wavelet, and eventually it gives you a matrix of transformation.

There are also other complex wavelet like *Daubechies Wavelets*. They all belong to *MRA (Multiresolution Analysis)*, and ideas are similar (have a mother fuction $\psi(x)$ and approximate function using $\psi(2^j x - k)$).

Fixed to Fixed Almost Lossless Coding

Let's look at our lossy coding techniques:

$$lossy\,coding\,techniques \begin{cases} trans form\,coding \\ rate - distortion\,theory \\ quantization \\ sub - bandcoding \end{cases}$$

In all of these cases, especially in transform coding, we are taking a set of vectors and transform it to another set of vector. So it is from fixed length message to fixed length codewords. No matter what the message is the length of codewords are the same.

But when we do lossless coding:

$$lossless\,coding\,techniques \begin{cases} Huf fman\,code \\ Lempel\,Ziv \end{cases}$$

They are all fixed to variable length codes. For example, when we do Huffman coding, if $\mathcal{X} = \{a, b, c, d\}$, our codewords can be $a = 0, b = 10, c = 110, d = 111$, which are not fixed.

Let assume $\mathcal{X}$ is a source alphabet, say {0,1}, and $x \in \mathcal{X}^n$. Can we uniformly map x to a k length string? Is there a way to do lossless coding in this case, which is fixed length to fixed length?

The answer is that we cannot do exact lossless coding but we can do something almost lossless. So that requires our decoder work with high probability.

Suppose we have a binary sequence of length n:

$$x = 00...101......$$

Assume each bit has $Pr(0) = 1 - p, Pr(1) = p$ and they are i.i.d.. Then we know:

$$Pr(00111011) = f(number\,of\,1's,\ L) = p^L(1-p)^{n-L}$$

So

$$Pr(a\,sequence\,having\,L\,1's) = \binom{n}{L} p^L(1-p)^{n-L} \approx \frac{2^{nh(\frac{L}{n})}}{\sqrt{n}} 2^{L\log p+(n-L)\log(1-p)}$$

$$= \frac{1}{\sqrt{n}} 2^{n[h(\frac{L}{n})+\frac{L}{n}\log p+(1-\frac{L}{n})\log(1-p)]}$$

$$= \frac{1}{\sqrt{n}} 2^{n[-\frac{L}{n}\log\frac{L}{np}-(1-\frac{L}{n})\log\frac{(1-\frac{L}{n})}{1-p}]}$$

$$= \frac{1}{\sqrt{n}} 2^{-nD(\frac{L}{n}||p)}$$

We know that $D(p||q) \geq 0$, and it is 0 only when $p = q$.

So when $L \neq np$, $D(\frac{L}{n}||p) > 0$, and $\frac{1}{\sqrt{n}}2^{-nD(\frac{L}{n}||p)}$ will diverge to 0 with $n \to \infty$. It is not 0 only when $L \approx np$, or $n(p - \epsilon) \leq L \leq n(p + \epsilon)$. Then in this case

$$\binom{n}{np} \sim 2^{nh(p)}$$

The size of these sequences is $2^{nh(p)}$, so we only need $nh(p)\,bits$ to keep any of them.

Suppose $k = nh(p)$, then we can have fixed length codes that use only $nh(p)$ bits. Therefore the rate of compression is

$$Rate = \frac{k}{n} = h(p) = entropy$$

Now we compress the data from fixed length to fixed length, but we have probability of error that equals to sequences that does not have np 1's. Even the probability is very small with very large n, those sequences are still there. But our decoder will work with very high probability. This connects lossless coding to lossy coding.

EE5585 Data Compression April 25, 2013

Lecture 25

Instructor: Arya Mazumdar *Scribe: Zhongyu He*

1 Wavelet Decomposition

Suppose we have a signal as the figure 1. What we want to do is to approximate the signal.

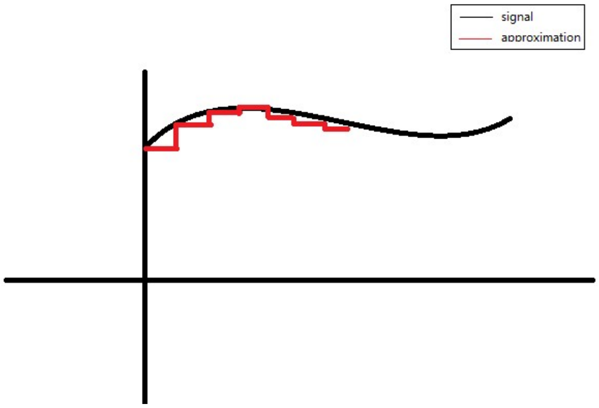


Figure 1: Approximated Signal.

One way of doing that is to take a function as figure 2 and use the shifted version of it. This is the simplest thing we can do. We can have this piecewise linear approximation as stated in red in figure 1. This also captures the quantization processes. The signal between two points will be quantized to a certain value.

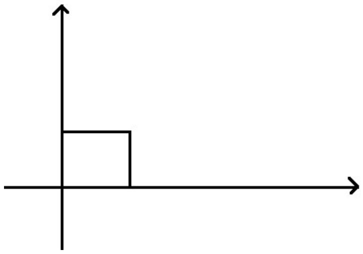


Figure 2: Block Wavelet

The signal in figure 2 is as following

$$\phi(t) = \begin{cases} 1 & 0 \le t < 1 \\ 0 & \texttt{otherwise} \end{cases}$$

The shifted version of this signal will be

$$\phi(t-k), k = 0, 1, \cdots$$

Using the shifted version of the function in figure 2, we can approximate the signal in figure 1 in some resolution. However, this function has very low resolution. Between 0 and 1 we have only 1 value. If we want to approximate the signal better, we need to scale the function.

The shifted scaled function will be as follows:

$$\phi(2t-k) = \begin{cases} 1 & 0 \le 2t-k < 1 \\ 0 & \texttt{otherwise} \end{cases} = \begin{cases} 1 & \dfrac{k}{2} \le t < \dfrac{k+1}{2} \\ 0 & \texttt{otherwise} \end{cases}$$

Now, with the new function above, we double the resolution. We can improve the resolution by scaling the function more as above. From the Nyquist rule, once the frequency is twice of the maximum frequency of the original signal, the signal can be recovered without any lose.

Now this function defined as:

$$\phi_{j,k}(t) = \phi(2^j t-k) = \begin{cases} 1 & 0 \le 2^j t-k < 1 \\ 0 & \texttt{otherwise} \end{cases} = \begin{cases} 1 & \dfrac{k}{2^j} \le t < \dfrac{k+1}{2^j} \\ 0 & \texttt{otherwise} \end{cases}$$

Use the above function to approximate the signal as stated in red in figure 1.

Then we have the 0^{th} order approximation. The coefficient C are average value of f between k and k+1.

$$\begin{aligned} \phi_f^0(t) &= \sum_k C_{0,k}\phi_{0,k}(t) \\ C_{0,k} &= \frac{1}{1}\int f(t)\phi_{0,k}(t)dt \\ &= \int_k^{k+1} f(t)dt \end{aligned}$$

This is actually a very low resolution approximation of the signal we have. In order to get high resolution, we can do higher order approximation.

1^{st} order approximation:

$$\begin{aligned} \phi_f^1(t) &= \sum_k C_{1,k}\phi_{1,k}(t) \\ C_{1,k} &= \frac{1}{1/2}\int f(t)\phi_{1,k}(t)dt \\ &= 2\int_{k/2}^{(k+1)/2} f(t)dt \\ C_{1,2k} &= 2\int_k^{k+1/2} f(t)dt \\ C_{1,2k+1} &= 2\int_{k+1/2}^{k+1} f(t)dt \\ C_{1,2k} + C_{1,2k+1} &= 2\int_k^{k+1} f(t)dt = 2C_{0,k} \\ \Rightarrow C_{0,k} &= \frac{1}{2}[C_{1,2k} + C_{1,2k+1}] \end{aligned}$$

We can go further and further. 2^{nd} order approximation:

$$
\begin{aligned}
\phi_f^2(t) &= \sum_k C_{2,k}\phi_{2,k}(t) \\
C_{2,k} &= \frac{1}{1/4}\int f(t)\phi_{2,k}(t)dt \\
&= 4\int_{k/4}^{(k+1)/4} f(t)dt \\
&\vdots \\
\phi_f^j(t) &= \sum_k C_{j,k}\phi_{j,k}(t) \\
C_{j,k} &= 2^j\int_{k/2^j}^{(k+1)/2^j} f(t)dt
\end{aligned}
$$

We can do the above procedures until we get the resolution we want. Suppose we are representing in 1^{st} order approximation.

$$\phi_f^1(t) = \sum_k C_{1,k}\phi_{1,k}(t)$$

Such representation can be split into two lower resolution parts:

$$\phi_f^0(t) = \sum_k C_{0,k}\phi_{0,k}(t), \quad C_{0,k} = \frac{1}{2}[C_{1,2k} + C_{1,2k+1}]$$

$$\texttt{and} \qquad \phi_f^1(t) - \phi_f^0(t)$$

We can see that higher resolution signal can be represented by lower resolution ones. Let's look at the interval from l to l+1.

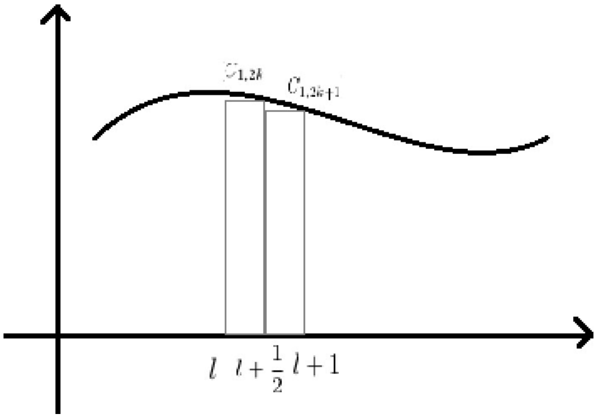


Figure 3: Interval from l to l+1.

We have:

$$\phi_f^1(t) - \phi_f^0(t) = \begin{cases} C_{1,2l} - C_{0,l} & l \le t < l + \frac{1}{2} \\ C_{1,2l+1} - C_{0,l} & l + \frac{1}{2} \le t < l+1 \end{cases} = \begin{cases} \frac{1}{2}[C_{1,2l} - C_{1,2l+1}] & l \le t < l + \frac{1}{2} \\ -\frac{1}{2}[C_{1,2l} - C_{1,2l+1}] & l + \frac{1}{2} \le t < l+1 \end{cases}$$

Define a function ψ:

$$\psi(t) = \begin{cases} 1 & 0 \le t < \frac{1}{2} \\ -1 & \frac{1}{2} \le t < 1 \end{cases}$$

Then we have

$$\begin{aligned} \phi_f^1(t) - \phi_f^0(t) &= \frac{1}{2}[C_{1,2l} - C_{1,2l+1}]\psi(t-l) \\ &= \sum b_{0,k}\psi_{0,k}(t) \\ \texttt{Where} \quad b_{0,k} &= \frac{1}{2}[C_{1,2k} - C_{1,2k+1}] \\ \psi_{0,k}(t) &= \psi(t-k) \end{aligned}$$

Now we can see that coefficients of lower resolution parts can be represented by coefficients of higher resolution parts. And the wavelet par $\psi(t)$ is called Haar Wavelet.

Depending on what kind of signal we want to approximate, blocks function may not be the best choice. Functions like figure 4 may be the optimal wavelet to approximate some signals.

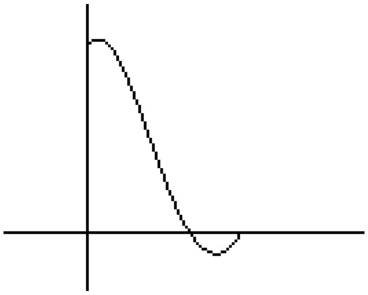


Figure 4: $\phi(t)$

Then we have shifted: $\phi(t-k)$, and scaled: $\phi(2^j t - k) = \phi_{j,k}(t)$. Then we will represent the signal with whatever resolution we need with these functions.

Again, as what we did above:

$$\begin{aligned} \phi_f^j(t) &= \sum C_{j,k}\phi_{j,k}(t) \\ C_{j,k} &= \frac{< f, \phi_{j,k} >}{< \phi_{j,k}, \phi_{j,k} >} \\ &= \frac{\int f(t)\phi_{j,k}(t)dt}{\int (\phi_{j,k}(t))^2 dt} \end{aligned}$$

Also $\phi_{j,k}(t)$ can be split into two parts. $\phi_{j-1,k}(t)$ and $\phi_{j,k}(t) - \phi_{j-1,k}(t)$. We can go further if $\phi(t)$ is known.

2 Data Compression Course Frame

What have we learn from this course? What to do with that? That is Data Compression. Data compression is not only compression but also recovery, because if you don't have a decoding algorithm, then you have no use to compress the data.

Therefore, there are two kinds of techniques base on the recovery that we can give. One is lossless recovery. The other one is lossy recovery.

2.1 Lossless Recovery

What is the limit of lossless compression? There are two parallel things.

Given a data string with unknown distribution, the best compression we can get is from ***Kolmogorov Complexity***. On the other hand, if the distribution of a random variable is known to us or we can try to model the data as a random variable, we can not go beyond the ***Entropy*** in compression.

So in some way, the Kolmogorov Complexity is more general than the Entropy.

Now there are several coding procedure that can take you to the value of Entropy. With these methods, we can get to the value of Entropy very closely, if not exactly.

2.1.1 variable length coding

1. Distribution is Known
Huffman Coding
Shannon Coding
Arithmetic Coding
In this part, we also have the notions of *Unique Decodability* and *Prefix free code.* All the codes are both unique decodable and prefix free code. We also discuss research problems on *fix free code.* 2. Universal Coding Lempel-Ziv Algorithm

2.1.2 Fixed to fix length coding

This is a special case when distortion is equal to zero.

2.2 Lossy Coding

More general case come from the Lossy coding when distortion is not zero. And the theory is given by ***Rate-Distortion***. We know this distortion can be the worst case and the average case. The Rate Distortion theory tells us that the compression algorithm we're using is lossy with some error. But given a certain amount of error, we can get the rate of compression we can achieve.

Then we look at ***Quantizers***. This is one important way to achieve this trade off between compression rate and rate distortion.

2.2.1 Scalar Quantizers

Uniform Quantizers
Optimal Quantizers.(We have to look at Lloyd's Algorithm)
In most of the cases, scalar quantizers are far away from what Rate Distortion theory promises. The solution is to go to Vector Quantizers.

2.2.2 Vector Quantizers

Basically in Vector Quantizers, we are doing ***clustering***.
We have multiple kinds of algorithms like ***Linde Buzo Gray*** and ***K-means-Clustering***.

There are two things the Vector Quantizers exploit.

1. Shape Gain

Most of the Vector Quantizers try to use rectangles to shape the decision region. This may not be the best way to quantize but it's the most prominent way in Vector Quantizers.

2. Correlation between Data.

We saw the example of Height-Weight. This example consist of random variables but they are highly correlated as in figure 5. We can draw a regress line and turn it into 1-dimensional quantization.

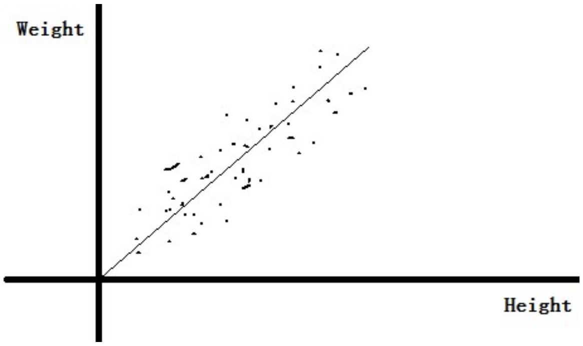


Figure 5: Height-Weight

When you have correlated data, one way to do that is to rotate the axes. That gives you another tool to go the rate-distortion functions apart from the quantizers.

2.2.3 Transform coding

In transform coding, you basically rotate the axes by using a linear operator. In this case, just multiply the data by a transform matrix to transform the data to another orthonormal axes.

1. Karhunen-Loeve transform

If we want to minimize the distortion, the best thing to do is Karhunen-Loeve transform, which depends on the data. The process of doing this transform is called Principle Component Analysis

If we want to find some algorithm that is independent of the data, we can look at the following.

2. DCT or DFT.

These are standard transform coding techniques working well for a very large classes of signals.

Now that we use transform coding to get rid of many dimensionality, why do we need to acquire that many signal? This leads to the idea of sensing of the signals. That's ***Compressed Sensing***.

3. Compressed Sensing.

This time, the transform matrix is not a square matrix but a rectangular matrix. So we need to look at the ***sufficient condition*** for existence of such sampling schemes. And then we have one decoding algorithm called ***Basic Pursuit*** that does the recovery. And there is Restricted Isometry Property for this scheme.

If we want to deal with a larger width of signals, we need some more techniques as follows.

4.Predictive Coding

5.Differential Coding(DPCM)

6.Subband Coding
We will separate out this different signals into wavelets. The most prominent wavelet is Haar Wavelet.

Above are the basic building blocks of this course. Most of the applications will be a combination of multiple techniques. They are highly correlated with each other.

3 Homework 4 Solution

Problem 1 Consider the following matrix ϕ:

$$\begin{bmatrix} 1 & 0 & 0.5 \\ -2 & 1 & 1 \end{bmatrix}$$

We observe samples of an 1-sparse vector x

$$y = \phi x = \begin{bmatrix} 0 \\ 5 \end{bmatrix}$$

Find out x.
Next, suppose we observe:

$$z = \phi x = \begin{bmatrix} 10.1 \\ -20.1 \end{bmatrix}$$

For a general vector x. Find out x using l1-minimization (basis pursuit).
Solution:
Question 1: Due to the linear independence, such problem has a unique solution. A universal way to solve is to assume:

$$x_1 = \begin{bmatrix} a \\ 0 \\ 0 \end{bmatrix}, x_2 = \begin{bmatrix} 0 \\ a \\ 0 \end{bmatrix}, x_3 = \begin{bmatrix} 0 \\ 0 \\ a \end{bmatrix}, \text{Then we will get the solution } x = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$$

Question 2: assume

$$x = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

$$\begin{aligned} min :& \|x\|_{l_1} \\ s.t. :& \phi x = z \\ \Rightarrow min :& |a_1| + |a_2| + |a_3| \\ s.t. :& a_1 + 0.5a_3 = 10.1 \\ & -2a_1 + a_2 + a_3 = -20.1 \\ \Rightarrow min :& |10.1 - 0.5a_3| + |0.1 - 2a_3| + |a_3| = f \end{aligned}$$

Case1: $a_3 > 20.2$

$$f = 0.5a_3 - 10.1 + 2a_3 - 0.1 + a_3 = 2.5a_3 - 10.2$$
$$\Rightarrow min.f = 2.5 \times 20.2 - 10.2 = 60.5, \quad when \quad a_3 = 20.2$$

Case2: $0.05 < a_3 < 20.2$

$$f = 10.1 - 0.5a_3 + 2a_3 - 0.1 + a_3 = 2.5a_3 + 10$$
$$\Rightarrow min.f = 2.5 \times 0.05 + 10 = 10.125, \quad when \quad a_3 = 0.05$$

Case2: $0 < a_3 < 0.05$

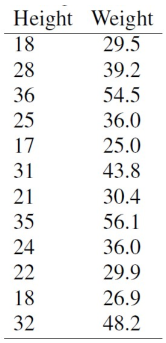
$$f = 10.1 - 0.5a_3 + 0.1 - 2a_3 + a_3 = -1.5a_3 + 10.2$$
$$\Rightarrow min.f = 10.125, \quad when \quad a_3 = 0.05$$

Case2: $a_3 < 0$

$$f = 10.1 - 0.5a_3 + 0.1 - 2a_3 - a_3 = -2.5a_3 + 10.2$$
$$\Rightarrow min.f = 10.2, \quad when \quad a_3 = 0$$

$$\Rightarrow x = \begin{bmatrix} 10.075 \\ 0 \\ 0.05 \end{bmatrix}$$

Problem 2 Following table shows height-weight data of 12 monkeys:



Height	Weight
18	29.5
28	39.2
36	54.5
25	36.0
17	25.0
31	43.8
21	30.4
35	56.1
24	36.0
22	29.9
18	26.9
32	48.2

Find out an orthonormal transform matrix to compress this data. Suppose you compress this two-dimensional data to one-dimensional. Show the data-compression procedure. What is the average error/distortion?

Solution:

The correlation of the data is y=1.5x. Then follow the procedure of transform coding to find angle θ and transform matrix

$$A = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}$$

Then follow the procedure and can easily get the solution.

EE5585 Data Compression May 2, 2013

Lecture 27

Instructor: Arya Mazumdar *Scribe: Fangying Zhang*

Distributed Data Compression/Source Coding In the previous class we used a H-W table as a simple example, now let's consider that as a X-Y table. The two columns are correlated. Suppose Alice has X and Bob has Y. Alice wants to compress X data by using entropy $H(X)$ and Bob wants to compress Y data by using entropy $H(Y)$. However they are using one decoder and the rate of compression is $R(X)+R(Y)=H(X)+H(Y)$. Now a remarkable thing about Slepian coding is even there's no link between Alice and Bob, they can still do better than $H(X)+H(Y)$. So what is the best thing you could do if **you know both X and Y** (X and Y is an entirety)? Here, you could do $H(X,Y)$,
$H(X,Y)=H(X)+H(Y|X)\leq H(X)+H(Y)$ (equal when they are iid.) so we can compress more than $H(X)+H(Y)$.

Let us formalize this is a more regress way. Now suppose you have two discrete sources X,Y and their distribution probability $p(x)$, $p(y)$, which are formed from making N independent drawings from a joint probability distribution $p(x,y)$. If you want to compress X, so you'll see a N-length sequence coming from X, which is expressed as X^n. And the encoding function would be mapping to a index 2^{nR_1} of a codeword:

$$f_1 : X^n \Rightarrow (1,2,...,2^{nR_1})$$

, then it needs $\log(2^{nR_1})=nR_1$ bits to express it. The Alice's compression rate is R_1 and so any n-bits sequence would be compressed to nR_1 bits. Similarly,

$$f_2 : Y^n \Rightarrow (1,2,...,2^{nR_2})$$

. The Bob's compression rate is R_2.

Then there is a single decoding function g, which is:

$$g : (1,2,...2^{nR_1}) * (1,2,...,2^{nR_2}) \Rightarrow x^n * y^n.$$

So this gives you back the sequences.
In this process, the error probability $Pr(g(f_1(x^n),f_2(y^n)\neq(x^n,y^n)))=P_e^{(n)}$. (R_1,R_2) is achievable if $\exists$ encoding and decoding functions, such that $P_e^{(n)}\Rightarrow 0$, and $R_1\geq H(X), R_2\geq H(Y)$.

Surprisingly, by using Slepian Wolf Theorem,
you can do as well as this with out communication between Alice and Bob.

1 Slepian Wolf Theorem

1973 Slepian Wolf Coding

The Slepian-Wolf theorem deals with the lossless compression of two or more correlated data streams (Slepian and Wolf, 1973). In the best-known variation, each of the correlated streams is encoded separately and the compressed data from all these encoders are jointly decoded by a single decoder as shown in Figure 1 for two correlated streams. Such systems are said to employ Slepian-Wolf coding, which is a form of distributed source coding. Lossless compression means that the source outputs can be constructed from the compression version with arbitrary small error probability by suitable choice of a parameter in the compression scheme.

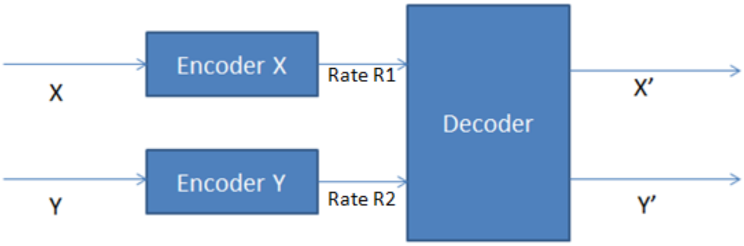


Figure 1: Block diagram for Slepian-Wolf coding: independent encoding of two correlated data streams X and Y

The efficiency of the system is measured by the rates in encoded bits per source symbol of the compressed data streams that are output by the encoders. The Slepian-Wolf Theorem specifies the set of rates that allow the decoder to reconstruct these correlated data streams with arbitrarily small error probability. As shown in Figure 1, encoder 1, observes X and then sends a message to the decoder which is a number from the set $1, 2, ..., 2^{nR_1}$. Similarly, encoder 2, observes Y and then sends a message to the decoder which is a number from the set $1, 2, ..., 2^{nR_2}$. The outputs from the two encoders are the inputs to the single decoder. The decoder, upon receiving these two inputs, outputs two n-vectors $\hat{X}$ and $\hat{Y}$ which are estimates of X and Y , respectively. The systems of interest are those for which the probability that $\hat{X}$ does not equal X or $\hat{Y}$ does not equal Y can be made as small as desired by choosing n sufficiently large. Such a system is said to be an **Achievable System** and the rate pair (R_1, R_2) for an achievable system is said to be an achievable rate pair. The achievable rate region is the closure of the set of all achievable rate pairs.

The achievable rate region for the pair of rates, (R_1, R_2) , is the set of points that satisfy the three inequalities:

$$1).R_1 + R_2 > H(X, Y)$$

$$2).R_1 > H(X|Y)$$

This means even if you've known Y(Alice knows Bob), X can not compress beyond $H(X|Y)$, and it's always smaller than $H(X)$. Same case for Bob.

$$3).R_2 > H(Y|X)$$

(2) and (3) declare that if there is a communication channel between Alice and Bob, or say they know each other what they have, the $H(X|Y)$ is the natural boundaries that they can know the best. This achievable rate region is shown in Figure 2.

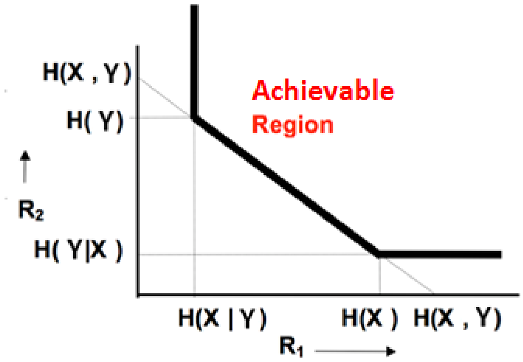
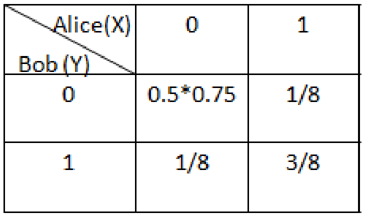


Figure 2: Achievable Rate Region

The significance of the Slepian-Wolf theorem is seen by comparing it with the entropy bound for compression of single sources. Separate encoders which ignore the source correlation can achieve rates only $R_1 + R_2 \geq H(X+Y)$. However, for Slepian-Wolf coding, the separate encoders exploit their knowledge of the correlation to achieve the same rates as an optimal joint encoder, namely, $R_1 + R_2 \geq H(X,Y)$.

2 Coin Tosses

Alice and Bob are tossing a coin.
For $x(Alice)andy(Bob) : Pr(1) = 0.5, Pr(0) = 0.5, Pr(x = y) = 0.75$. They are correlated. This table gives a $p(x,y)$.



Alice(X) / Bob (Y)	0	1
0	0.5*0.75	1/8
1	1/8	3/8

Figure 3:

Now Alice and Bob want to compress this tosses sequence. Because this is half-half probability, they both need 1bit at least, which is 2bits totally to store. However, now if you use Slepian, you can compress more than this. You can achieve:

$$\begin{aligned}
R_1 + R_2 &= H(X,Y) \\
&= \frac{3}{8} * \log(\frac{8}{3}) + \frac{1}{8} * \log 8 + \frac{3}{8} * \log(\frac{8}{3}) + \frac{1}{8} * \log 8 \\
&= 3 - \frac{3}{4} * \log 3 \\
&= 1.811 \\
R_1 &\geq H(X|Y) \\
&= h(\frac{3}{4})
\end{aligned}$$

$$\begin{aligned}
&= \frac{3}{4} * \log(\frac{4}{3}) + \frac{1}{4} * \log 4 \\
&= 2 - 1.189 \\
&= 0.811.
\end{aligned}$$

Similarly, $R_2 \geq 0.811$. So you can still compress more even there is no communication between X and Y.

3 Proof: Typical Sequence

Before we prove this, let's recall if we have only on single source. We call this a fixed-to-fixed N-length coding, which depends on the typical sequence suppose you have $x^n \in X^n$ is called typical sequence if $2^{-n(H(X)+\epsilon)} \leq Pr(x^n) \leq 2^{-n(H(X)-\epsilon)}$. If $A_\epsilon^{(n)}$ is a set of typical sequence, then
$2^{-n(H(X)-\epsilon)} \leq |A_\epsilon^{(n)}| \leq 2^{-n(H(X)+\epsilon)}$
This is called **Asymmetric Equiputation Property**. If you want to get a set of typical sequence, you need to log the left and right side.

4 Another Proof: Random Binning

If we have $f : x_n \Rightarrow 1, 2, ..., 2^{nR}$, then $f(x^n) =$ Random index from $1, 2, ..., 2^{nR}$. Consider $g : 1, 2, ..., 2^{nR} \Rightarrow x^n$, which means find out a typical sequence that was signal to the index i and output that

$$\begin{aligned}
P_e^{(n)} &= Pr(x^n \notin A_\epsilon^{(n)}) + Pr(\exists y^n \neq x^n, y^n \in A_\epsilon^{(n)}, f(y^n) = f(x^n)) \\
&= \sum_{x^n \in X^n} P(x^n) Pr(\exists y^n \in A_\epsilon^{(n)}; f(y^n) = f(x^n)) \\
&= 3 - \frac{3}{4} * \log 3 \\
&\leq \sum_{x^n \in X^n} P(x^n) \sum_{y^n \in A_\epsilon^{(n)} : y^n \neq x^n} Pr((y^n) = f(x^n)) \\
&\leq \sum_{x^n \in X^n} P(x^n) \sum_{y^n \in A_\epsilon^{(n)}} 2^{-nR} \\
&= 2^{-nR} 2^{n(H(X)+\epsilon)}
\end{aligned}$$

Whenever $R > H(X) + \epsilon$, $2^{n(H(X)+\epsilon)} \Rightarrow 0$. So we can get a code arbitrarily close to $H(X)$. Now we need to extend this idea for that distributed.

What we have here two encoders:

$$f_1 : x^n \Rightarrow 1, 2, ..., 2^{nR_1}$$

and

$$f_2 : y^n \Rightarrow 1, 2, ..., 2^{nR_2}.$$

You can consider each of the index of bins, so whenever you have a sequence, you randomly throw it in one of the bins. f_1 and f_2 are both random binning. The idea is the sequence we have is a typical sequence but no bin will contain more than one typical sequence, cuz there's no enough number of bits. So only by looking at the bin, you can find out what the sequence was. The decoder $g(i, j)$: Find out $(x^n, y^n) \in A_\epsilon^{(n)}$, such that $f_1(x^n) = i$ and $f_2(y^n) = j$.
When will this be successful? Again, let's calculate the error $P_e^{(n)}$.

$$\begin{aligned}
E_0 &= (x^n, y^n) \notin A_\epsilon^{(n)} \\
E_1 &= \exists \tilde{x^n} \neq x^n, (\tilde{x^n}, y^n) \in A_\epsilon^{(n)}); f_1(\tilde{x^n}) = f_1(x^n) \\
E_1 &= \exists \tilde{y^n} \neq y^n, (\tilde{x^n}, y^n) \in A_\epsilon^{(n)}); f_2(\tilde{y^n}) = f_2(y^n) \\
E_2 &= \exists (\tilde{x^n}, \tilde{y^n}), \tilde{x^n} \neq x^n, \tilde{y^n} \neq y^n, (\tilde{x^n}, \tilde{y^n}) \in A_\epsilon^{(n)}); f_1(\tilde{x^n}) = f_1(x^n), f_2(\tilde{y^n}) = f_2(y^n) \\
P_e^{(n)} &= Pr(E_0 \cup E_1 \cup E_2 \cup E_{12}) \\
&\leq Pr(E_0) + Pr(E_1) + Pr(E_2) + Pr(E_{12}) > 0, \\
Pr(E_1) &= \sum_{(} x^n, y^n) P(x^n, y^n) Pr(\exists \tilde{x^n}, (\tilde{x^n}, y^n) \in A_\epsilon^{(n)}, f^1(\tilde{x^n} = f^1(x^n))) \\
&\leq \sum_{(} x^n, y^n) P(x^n, y^n) \sum_{(} \tilde{x^n}, (\tilde{x_1^n}, \tilde{y^n}) \in A_\epsilon^{(n)}) 2^{-nR_1} \\
&= 2^{-nR_1} A_\epsilon^{(n)}(X|Y) \\
&\leq 2^{-nR_1} * 2^{n(H(X|Y)+\epsilon}. \\
E_0 &\Rightarrow 0;
\end{aligned}$$

When $R_1 > H(X|Y) + \epsilon$, $Pr(E_1) \Rightarrow 0$; when $R_2 > H(Y|X) + \epsilon$, $Pr(E_2) \Rightarrow 0$; when $R_1 + R_2 > H(X, Y)$, $Pr(E_{12}) \Rightarrow 0$; Thus $P_e^{(n)} = 0$.

What is the more efficient encoder would be?
If $X \sim Ber(\frac{1}{2})$, $Y \sim Ber(\frac{1}{2})$, $Pr(x = y) = \frac{3}{4}$. X and Y are correlated. Suppose $Pr(x \neq y)) = p$, $Pr(x = y) = 1 - p)$. $p = \frac{1}{4}$ in this case. Alice(X) compresses up to H(X), which is 1bit. Bob(Y) will compress in the following way. He will take a random Bernoulli $m * n (m < n)$ matrix Φ multiply the y sequence and get a real integer number set. Let this number set mod2, which means if you have even number, you place a 0; if you have a odd number, you place a 1. Eventualy you get n bits by this multiplication and totally you have $1 + \frac{m}{n}$ bits. So the compression rate is $1 + \frac{m}{n}$.
Now there are two things we need to know. First of all is what is the value of m from Y. This a problem from a $m * n$ bits to m bits. The following method can solve it.

Step: 1 Decode to find x^n
Once you find x^n, the unknown is y^n.

$$\begin{aligned}
\Phi x^n \bmod 2 &\Rightarrow \Phi y^n \bmod 2 \\
&= \Phi(x^n - y^n) \bmod 2 \\
&= \Phi z^n \bmod 2
\end{aligned}$$

Here,

$$z^n = z_1 z_2 ... z_n = x^n - y^n$$

Each of there z are iid.

$$Pr(z^i = 1) = Pr(x_i \neq y_i) = p,$$
$$Pr(z^i = 0) = 1 - p.$$

Then find out z^n from $\Phi z^n mod 2$. Once you find out z^n, you can add it by x^n and thus you get y^n. The condition of finding this z^n :
The typical set of z is

$$2^{-n(H(X)-\epsilon)} \leq |A_\epsilon^{(n)}(z)| \leq 2^{-n(H(X)+\epsilon)}$$

in which,

$$H(z) = H(p) = -p \log p - (1-p) \log(1-p)$$

and,

$$|A_\epsilon^{(n)}(z)| \sim 2^{nh(p)}$$

$\exists$ a typical $z\tilde{(}n) \in A_\epsilon^{(n)}(z)$ for which the probability

$$Pr(\Phi \tilde{z^n} = \Phi z^n \text{ mod } 2)$$

The error probability

$$P_e^{(n)} \leq 2^{nh(p)} * 2^{-m}$$

if the error $m > nh(p)$, then $\frac{m}{n} > h(p) + c$ and $P_e^{(n)}$ would be zero.

EE5585 Data Compression May 7, 2013

Lecture 28

Instructor: Arya Mazumdar *Scribe: Yijun Lin*

1 Distributed Data Compression

From the previous lectures we have learn that for two data sources X and Y, one can acheive rates at $R_1 + R_2 \geq H(X + Y)$, with two seperated encoders which ignore the source correlation. However, if the two sources X and Y are correlated with each other, an optimal joint encoder can acheive compression rates at H(X,Y), by applying the Slepian Wolf Theorem.

Now let us consider a similar problem, supoose there are two set of data X and Y, which both have n number of elements (x1, x2, x3...xn) and (y1, y2, y3,... yn). These two set of data are corelated and not independent. The data sets are sent to two different encoders before being compressed together and being sent to a single joint decoder, as the relation shown in Figure 1.

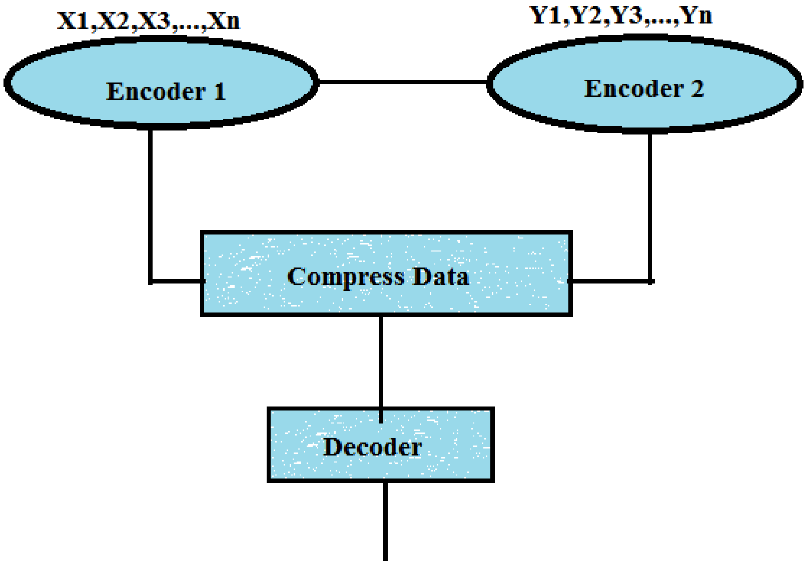


Figure 1: Block diagram of two correlated data sets

Such system is measured by the compress rate in bits per source symbol of the output streams of encoders. The single decoder is designed to be able to reconstruct the correlated data streams after compression in an optimal way. As we can see, encoder 1 has a rate of R1 = nH(X) while encoder 2 has a rate of R2 = nH(Y). The claim is if there is a communication channel between Encoder 1 and Encoder 2, by Slepian Wolf Theorem, the achievable rate of compression is:

$$nH(X) + nH(Y) \geq nH(X,Y) \qquad \cdots\cdots \quad (1)$$

which means it can be compressed to H(X,Y) bits/symbol.

1.1 Distributed Data Compression with Bernoulli Source

Now let us consider the following example: assume there are two correlated data sources X ($x_1,x_2,x_3...x_n$) and Y($y_1,y_2,y_3...y_n$). X is Bernoulli(q) with the following properties:

$$\Pr(xi = yi) = 1- p; \quad \Pr(xi \neq yi) = p.$$

Hence, Y is also Bernoulli(p + q +2pq) since:

$$\begin{aligned} \Pr(Y=1) &= \Pr(X = 1 \text{ and } Y = X) + \Pr(X = 0 \text{ and } Y \neq X) \\ &= q (1 - p) + (1 - q)p \\ &= p + q + 2pq \end{aligned}$$

The rate of compression we can achieve is to equal to: *nh(q) + nh(p + q +2pq)* bits. However, by the inequality (1) the optimal solution is to achieve *nh(q) + nh(p)* bits.

Since:

$$\begin{aligned} n[H(X,Y)] &= n[H(X) + H(Y|X)] \\ &= n[H(X) + H(Z)] \\ &\qquad \text{Where } Z_i = X_i + Y_i \bmod 2, \text{ for } i = 1 \text{ to } n \\ &= n[h(q) + h(p)] \end{aligned}$$

Notice it is always true that $p < \frac{1}{2}$, implies that:

$$q (1 - 2p) > 0 \qquad \cdots\cdots\cdots\cdots>> \qquad p + q +2pq > p$$

and *nh(q) + nh(p)* is a increasing function.

1.2 Distributed Data Compression over Mod-2 Arithmetic

Suppose there are two data sources X ($x_1,x_2,x_3...x_n$) and Y($y_1,y_2,y_3...y_n$) over vector space $F^n = \{0, 1\}^n$. Let Z = X + Y. Notice that Z is also equal to X - Y or Y - X, due to fact that they are in mod-2 arithmetic.

Figure 2 shows the block diagram of sources X, Y and Z, where Z is Bernoulli(p).

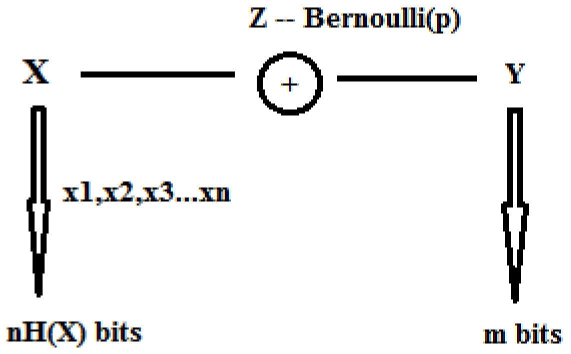


Figure 2

The rates of compression of X and Y is nH(X) and m bits, respectively. From what was discussed above, the achievable number of bits per symbol for such problem is:

$$H(X) + \frac{m}{n} = h(q) + \frac{m}{n}$$

Then we claim that there exists an m×n (m < n) matrix Φ, such that m ≈ *nh(p)*. With such matrix Φ, source data Y can be recovered from ΦY.

Here is the process of how to decode in the decoder: when we looking at the output of decoder, we have nH(X) and ΦY. Then do the following:

(1) decode X from nH(X);

(2) find ΦX simply by multiplying with each other;

(3) find ΦZ = Φ(X+Y) = ΦX + ΦY;

(4) recover Z from ΦZ;

(5) find Y from Y = X + Z.

However, when recovering Z from ΦZ from step (4), there is no unique solution Z. But since any solution of Z would work for the problem purpose, we turn to the typical set of Z, with arbitrarily small error probability. First, let us put some definitions around here:

We know that Z is a Bernoulli(p), let ΦZ = b for simple. For the typical set of Z: $A_\varepsilon^{(n)}(p)$ and $| A_\varepsilon^{(n)} | \le 2^{n(h(p)+\varepsilon}$. Now the goal is to find a vector $\hat{Z}$ from the typical set $A_\varepsilon^{(n)}$ that satisfies $\Phi\hat{Z} = b$. Such job could be complicated and require computer as a companion. But once we have the output $\hat{Z}$ from whatever the tools we may used, we can recover Y from $Y = X + \hat{Z}$.

The probability of error $P_e^{(n)}$can be calculated in bounds of:

$$P_e^{(n)} \le \Pr(Z \notin A_\varepsilon^{(n)}(p)) + \Pr(\exists Z' \neq Z;\ Z \in A_\varepsilon^{(n)}(p) \text{ and } \Phi Z' = b)$$

$$\le \sum_{Z' \in A\varepsilon(n)(p) \text{and } Z' \neq Z} \Pr(\Phi Z' = b)$$

To calculate Pr ($\Phi Z' = b$), we first choose Φ to be randomly uniformly with iid Bernoulli($\frac{1}{2}$) entries.

$$\begin{bmatrix} \Phi 11 & \cdots & \Phi 1n \\ \vdots & \ddots & \vdots \\ \Phi m1 & \cdots & \Phi mn \end{bmatrix} \times \begin{matrix} Z1' \\ \vdots \\ Zn' \end{matrix} = \begin{matrix} b1 \\ \vdots \\ bm \end{matrix}$$

Φ matrix: m× n Z' b

$\Phi_{11}Z_1' + \Phi_{12}Z_2' + \cdots + \Phi_{1n}Z_n' = b_1$

Since Φ is Bernoulli(1/2), the probability of making Z' to equal b_i is equal to $\frac{1}{2}$, implies that:

Pr ($\Phi Z' = b$) $= \frac{1}{2} \times \frac{1}{2} \times \cdots \times \frac{1}{2} = \frac{1}{2^m}$. Substitute it back the error inequality:

$$P_e^{(n)} \leq | A_\varepsilon^{(n)}(p) | \times \frac{1}{2^m}$$

$$\leq 2^{n(h(p)+\varepsilon)} \times \frac{1}{2^m}$$

$P_e^{(n)}$ goes to $\frac{1}{2}$nε' if $m = n(h(p) + \varepsilon + \varepsilon'$.

Now that $nh(q) + m = n[h(q) + \frac{m}{n}] \approx n[h(q) + h(p)]$.

1.3 Distribution Data Compression when two sources are closed

Consider a similar problem with the case that the two sources X ($x_1,x_2,x_3...x_n$) and Y($y_1,y_2,y_3...y_n$) are very closed to each other (ie. X-Y to be minimum). The process diagram is shown in Figure 3 below:

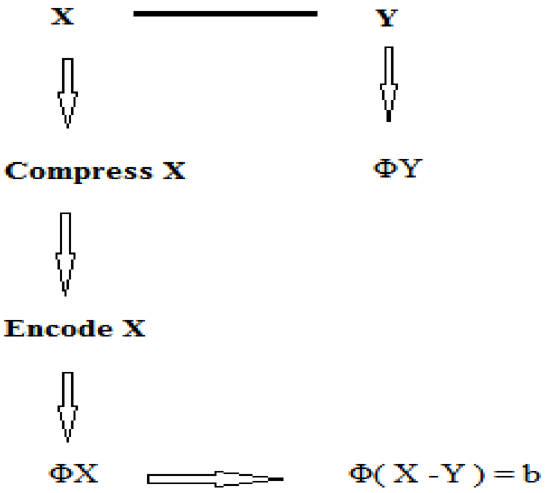


Figure 3

Let Z = X - Y, which can be considered as a noise function $\mathcal{N}(0, \sigma^2)$. The job once again is to find Z from ΦZ = b. Since the noise figure should be as small as possible, we want to find min $||Z||_{l2}^2$, such that ΦZ = b.

♠ ***Claim:*** Assuming that $\Phi\Phi^{-1}$ is nonsingular, $\hat{Z} = \Phi^T (\Phi\Phi^T)^{-1} b$

♠ ***Proof:*** suppose that Z satisfies ΦZ = b.

$$\text{Then } ||Z||_{l2}^2 = ||Z - \hat{Z} + \hat{Z}||_{l2}^2$$

$$= ||Z - \hat{Z}||_{l2}^2 + Z\,\hat{Z}^T(Z - \hat{Z}) + ||Z||_{l2}^2$$

The term of $Z\,\hat{Z}^T(Z - \hat{Z})$ goes to zero due to:

$$\hat{Z}^T(Z - \hat{Z}) = b^T(\Phi\Phi^T)^{-1}\Phi(Z - \hat{Z})$$

$$= b^T(\Phi\Phi^T)^{-1}(b - b)$$

$$= 0$$

Hence, $||Z||_{l2}^2 \geq ||\hat{Z}||_{l2}^2$

Now we have: $\Phi\hat{Z} = \Phi\Phi^T (\Phi\Phi^T)^{-1} b$, which could be used as a solution.

2 Linear Error-Correcting Code

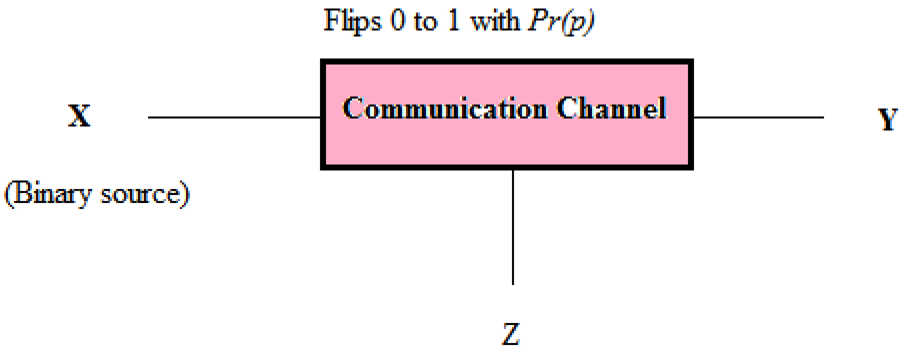


Figure 4

In general, as Figure 4 shown, suppose we have a data sources $X \in F^n = \{0, 1\}^n$ going through a communication channel, which flips 0 to 1 with probability of Pr(p),with some noises Z. the output Y of the channel is known. The process we discussed above is used to recover X by finding Z and calculating ΦY. Such processing code is called Linear Error-Correcting code.