EE5585 Data Compression	February 28, 2013
Lecture 11	
Instructor: Arya Mazumdar	*Scribe: Nanwei Yao*

Rate Distortion Basics

When it comes to rate distortion about random variables, there are four important equations to keep in mind.

1. The entropy
$$H(X) = \sum_{x \in \mathcal{X}} p(x) \log p(x)$$
2. Conditional entropy
$$H(X|Y) = -\sum_{x,y} p(x,y) \log p(x|y)$$
3. Joint entropy
$$H(X,Y) = \sum_{x \in \mathcal{X}} p(x) \log p(x)$$
4. Mutual Information
$$I(X;Y) = H(X) - H(X|Y)$$

We already know from previous lecture that

$$H(X,Y) = H(X) + H(Y|X)$$

But previous proof is a little bit complex, thus we want to prove this equation again in a simpler way to make it clearer.

Proof:

$$\begin{aligned}
H(X,Y) &= -\sum p(x,y) \log p(x,y) \\
&= -\sum p(x,y) \log[p(x)p(y|x)] \\
&= -\sum p(x,y) \log p(x) - \sum p(x,y) \log p(y|x) \\
&= -\sum_x \log p(x) \sum_y p(x,y) + H(Y|X) \\
&= -\sum_x \log p(x) \sum_x p(x) + H(Y|X) \\
&= -\sum_x p(x) \log p(x) + H(Y|X) \\
&= H(X) + H(Y|X)
\end{aligned}$$

Definition:For random variables (X,Y) whose probabilities are given by p(x,y), the *conditional entropy* $H(Y|X)$ is defined by

$$H(Y|X) = \sum_{x \in \mathcal{X}} p(x) H(Y|X = x)$$

$$= -\sum_{x\in\mathcal{X}} p(x) \sum_{y\in\mathcal{Y}} p(y|x) \log p(y|x)$$
$$= -\sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x,y) \log(p(y|x)$$

One important thing to notice is that $H(X) \geq H(X|Y)$, this means that conditioning always reduces the entropy. The entropy of a pair of random variables is the summation of the entropy of one plus the conditional entropy of the other. This is based on *Chain rule*:

$$H(X,Y) = H(X) + H(Y|X)$$

Thus, $I(X;Y) = H(X) + H(Y) - H(X,Y)$. We can understand the mutual information $I(X;Y)$ as the reduction in the uncertainty of X because of some of our knowledge of Y. By symmetry, we also have $I(X;Y) = I(Y;X)$. Thus we know that the information X provides us has the "same amount" that Y provides us.

Rate Distortion New Materials

Recall that in *lossy coding*, we cannot compress a file without error, and we want the average distortion to be bounded above. For a binary file which is of our interest, we use *Hamming Distance* (probability of error distortion) for distortion function.
Another important case is Quantization. Suppose we are given a Gaussian random variable, we quantize it and represent it by bits. Thus we lose some information. What is the best Quantization level that we can achieve? Here is what we do.
Define a random variable $X \in \mathcal{X}$. Our source produces a n length vector and we denote it by $X^n = X_1, X_2, ..., X_n$, where the vector is i.i.d. and produced according to the distribution of random variable X and $p(x) = Pr(X = x)$. What we do is to encode the file. After the encoder, the function f_n gives us a compressed string. Then we map the point in the space to the nearest codeword and we obtain an index which is represented by $\log M_n$ bits. Finally, we use a decoder to map the index back which gives us one of the codeword $\hat{x}$ in the space.

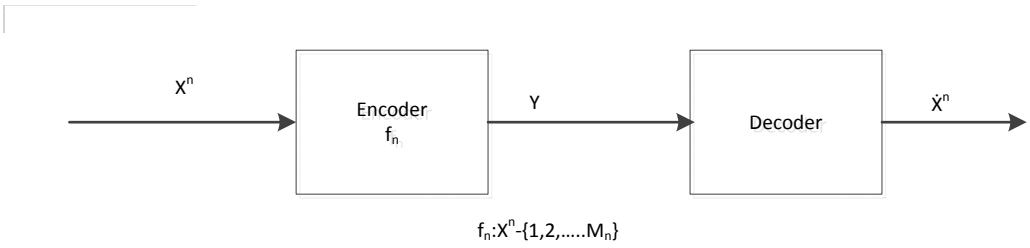

Figure 1: Rate Distortion Encoder and Decoder

Above is a figure of rate distortion encoder and decoder. Where $f_n = X^n \to \{1, 2, ...M_n\}$ is the encoder, and $\hat{X}^n$ is the actual code we choose.

Definition: The *distortion between sequences* x^n and $\hat{x^n}$ is defined by

$$d(x^n, \hat{x}^n) = \frac{1}{n} \sum_{i=1}^{n} d(x_i, \hat{x}_i) \tag{1}$$

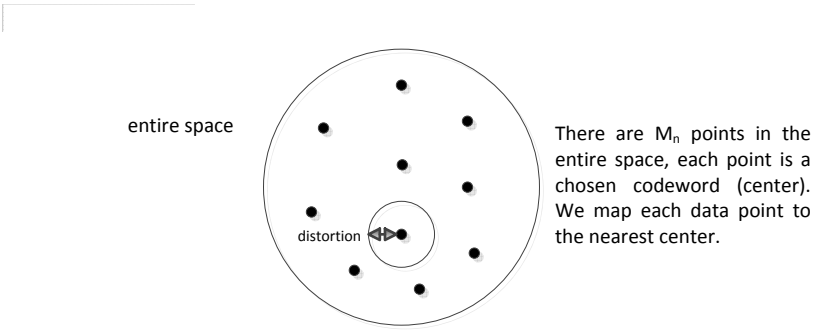

Figure 2: Codewords and Distortion

Thus, we know that the distortion of a sequence is the average distortion of the symbol-to-symbol distortion. We would like $d(x^n, \hat{x}^n) \leq D$. The compression rate we could achieve is $R = \lim_{x \to +\infty} \frac{\log M_n}{n}$ bits/symbol.The *Rate distortion* function $R(D)$ is the minimization of $\frac{\log M_n}{n}$ such that $Ed(x^n, \hat{x}^n) \leq D$ at the limit of $n \to \infty$.

Theorem(Fundamental theory of source coding): The *information rate distortion function* $R(D)$ for a source X with distortion measure $d(x, \hat{x})$ is defined as

$$R(D) = \min_{p(\hat{x}|x): \sum_{x,\hat{x}} p(x)p(\hat{x}|x)d(x,\hat{x}) \leq D} I(X; \hat{X})$$

From the above equation, we could see that no n involved. This is called a single letter characterization. For the subscript of the summation, we know that $\sum p(x, \hat{x})d(x, \hat{x}) = \sum_{x,\hat{x}} p(x)p(\hat{x}|x)d(x, \hat{x}) \leq D$.
To prove this theorem, we want to show $R(D) \geq \min_{p(\hat{x}|x): \sum_{(x,\hat{x})} p(x)p(\hat{x}|x)d(x,\hat{x}) \leq D} I(X; \hat{X})$ first, and the $R(D) \leq \min_{p(\hat{x}|x): \sum_{(x,\hat{x})} p(x)p(\hat{x}|x)d(x,\hat{x}) \leq D} I(X; \hat{X})$ will be shown in the next lecture.

First of all, let's see what is *Chain Rule*. It is defined as below:

$$H(X_1, X_2, X_3, ..., X_n) = H(X_1) + H(X_2|X_1) + H(X_3|X_1, X_2) + ... + H(X_n|X_1, X_2, ..., X_{n-1})$$

Chain rule can be easily proved by *induction*.
Besides chain rule, we also so need the fact that

$$R(D) = \min_{p(\hat{x}|x): \sum_{(x,\hat{x})} p(x)p(\hat{x}|x)d(x,\hat{x}) \leq D} I(X; \hat{X})$$

is convex.

Two Properties of R(D)

We now show two properties of R(D) that are useful in proving the converse to the rate-distortion theorem.

1. R(D) is a decreasing function of D.
2. R(D) is a convex function in D.

For the first property, we can prove its correctness intuitively: If R(D) is an increasing function, this means that the more the distortion, the worse the compression. This is definitely what we don't want.

Now, let's prove that second property.

Proof: Choose two points $(R1, D1), (R2, D2)$ on the boundary of R(D) with distributions $P_{\hat{X}_1|X}$ and $P_{\hat{X}_2|X}$. Then, we can construct another distribution $P_{\hat{X}_\lambda|X}$ such that

$$P_{\hat{X}_\lambda|X} = \lambda P_{\hat{X}_1|X} + (1-\lambda)P_{\hat{X}_2|X}$$

where $0 \leq \lambda \leq 1$.The average distortion D_λ can be given as

$$\begin{aligned} EP_{X,\hat{X}_\lambda}[d(X,\hat{X}_\lambda)] &= \lambda EP_{X,\hat{X}_1}[d(X,\hat{X}_1)] + (1-\lambda)EP_{X,\hat{X}_2}[d(X,\hat{X}_2)] \\ &= \lambda D_1 + (1-\lambda)D_2 \end{aligned}$$

We know that $I(X;Y)$ is a convex function of $p_{\hat{X},X}()$ for a given $p_X()$. Therefore,

$$I(\hat{X}_\lambda; X) \leq \lambda I(\hat{X}_1; X) + (1-\lambda)I(\hat{X}_2; X)$$

Thus,

$$\begin{aligned} R(D_\lambda) = I(\hat{X}_\lambda) &\leq \lambda I(\hat{X}_1; X) + (1-\lambda)I(\hat{X}_2; X) \\ &= \lambda R(D_1) + (1-\lambda)R(D_2) \end{aligned}$$

Therefore, R(D) is a convex function of D.

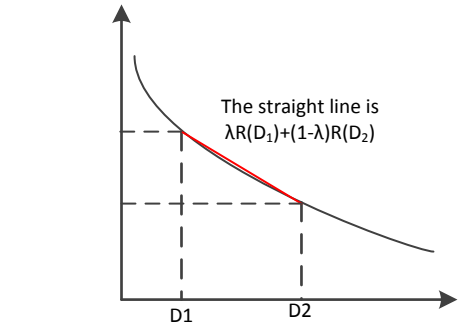

Figure 3: Function of $R(D)$ and $R(\hat{D})$

For the above proof, we need to make the argument that distortion D is a linear function of $p(\hat{x}|x)$. We know that the D is the expected distortion and it is given as $D = \sum p(x,\hat{x})d(\hat{x},x) = \sum p(x)p(\hat{x}|x)d(x,\hat{x})$. If we treat $p(\hat{x}|x)$ as a variable and both $p(x)$ and $d(x,\hat{x})$ as known quantities, we know that D is a linear function of $p(\hat{x}|x)$. Therefore, $R(D)$ is a convex function of $p(\hat{x}|x)$. The proof that $I(X;\hat{X})$ is a convex function of $p(\hat{x}|x)$ will not be shown here.

Converse Argument of R(D)

The converse argument of R(D) tells us that for any coding scheme whose expected distortion is at most to be D, there doesn't exist a code such that its rate is less than R(D). Now, let's prove it.
Proof

$$\begin{aligned}
\log M_n &\geq H(\hat{X}^n) \\
&\geq H(\hat{X}^n) - H(\hat{X}^n|X^n) \\
&= I(\hat{X}^n; X^n) \\
&= H(X^n) - H(X^n|\hat{X}^n) \\
&= \sum_{i=1}^{n} H(X_i) - \sum_{i=1}^{n} H(X_i|\hat{X}^n, X_1, X_2, ...X_n) \\
&\geq \sum_{i=1}^{n} H(X_i) - \sum_{i=1}^{n} H(X_i|\hat{X}_i) \\
&= \sum_{i=1}^{n} I(X_i; \hat{X}_i)
\end{aligned}$$

Recall that $R(D) = \min_{p(\hat{x}|x):\sum_{(x,\hat{x})} p(x)p(\hat{x}|x)d(x,\hat{x}) \leq D} I(X; \hat{X})$, thus

$$\begin{aligned}
\log M_n &\geq \sum_{i=1}^{n} I(X_i; \hat{X}_i) \\
&\geq \sum_{i=1}^{n} R(Ed(X_i; \hat{X}_i) \\
&= n \sum_{i=1}^{n} \frac{1}{n} R(Ed(X_i; \hat{X}_i)) \\
&\geq nR(\frac{1}{n} \sum_{i=1}^{n} Ed(X_i; \hat{X}_i)) \\
&= nR(\frac{1}{n} E[\sum_{i=1}^{n} d(X_i, \hat{X})]) \\
&\geq nR(D)
\end{aligned}$$

We see from the proof that $R = \lim_{x \to +\infty} \frac{\log M_n}{n} \geq R(D)$, thus, our proof is finished.

Example of Rate Distortion Theorem

An interesting example to look at is a binary file. What is $R(D)$ for a given binary file?
We already know from previous lectures that $R(D) = 1 - H(D)$. But this equation is too general to use for our given file.
So given a source $\mathcal{X} = \{0, 1\}$, suppose X has a Bernoulli(p) distribution, i.e. $Pr(X = 1) = p$ and $Pr(X = 0) = 1 - p$.Then

$$\begin{aligned}
I(x; \hat{x}) &= H(X) - H(X|\hat{X}) \\
&= h(p) - H(X|\hat{X})
\end{aligned}$$

$$
\begin{aligned}
&= h(p) - H(X \oplus \hat{X}|\hat{X}) \\
&\geq h(p) - H(X \oplus \hat{X}) \ (conditionality\, reduces\, entropy) \\
&= h(p) - H(Y)
\end{aligned}
$$

where $Y = X \oplus \hat{X}$. It is clear that $Pr(Y = 1) = Pr(X \neq \hat{X})$, and

$$
\begin{aligned}
\sum p(X, \hat{X}) d(X, \hat{X}) &= p(0,1) + p(1,0) \\
&= Pr(X \neq \hat{X}) \\
&= Pr(Y = 1) \leq D
\end{aligned}
$$

Recall from previous lectures, for any $D \leq \frac{1}{2}$, the binary entropy function $h(D)$ is increasing. Thus $h(p) - H(Y) \geq h(p) - h(D)$ for $D \leq \frac{1}{2}$. We have showed that $R(D) \geq h(p) - h(D)$, and we will show $R(D) = h(p) - h(D)$. When $p = \frac{1}{2}$, $R(D) = 1 - h(D)$.

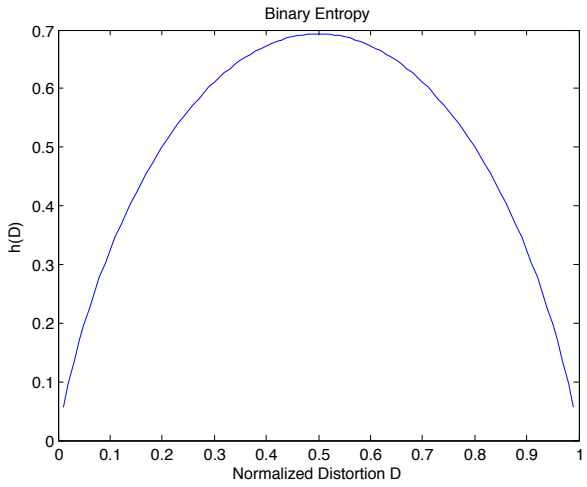

Figure 4: Binary Entropy

Up to this point, we want to show $H(X \oplus \hat{X}|X)$ has the same value as $h(D)$.

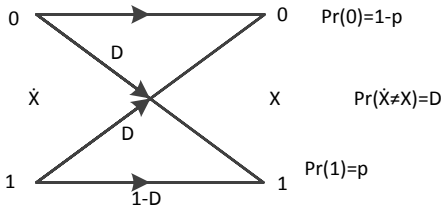

Figure 5: Binary Encoding Demonstration

We know that $Pr(\hat{X} \neq X) = D$. Assume $Pr(\hat{X} = 0) = 1 - r$ and $Pr(\hat{X} = 1) = r$, so $Pr(X = 0) = (1 - r)(1 - D) + rD = 1 - p$. Solve this equation for r, we obtain $r = \frac{p - D}{1 - 2D}$ for $D \leq \frac{1}{2}$.

In this case, $I(X;\hat{X}) = H(X) - H(Y) = h(p) - h(D)$, thus we proved both sides of the main theory for binary source.

Midterm Solutions

1. $1 - h(D)$ is the optimal rate of compression that is achievable.$1 - h(D) = 1 - h(1/20)$, where $h(x) = -x \log x - (1-x)\log(1-x)$.

2. $\frac{\pi(\frac{1}{4})^2}{(\frac{1}{2})^2} = \frac{1}{4}\pi$

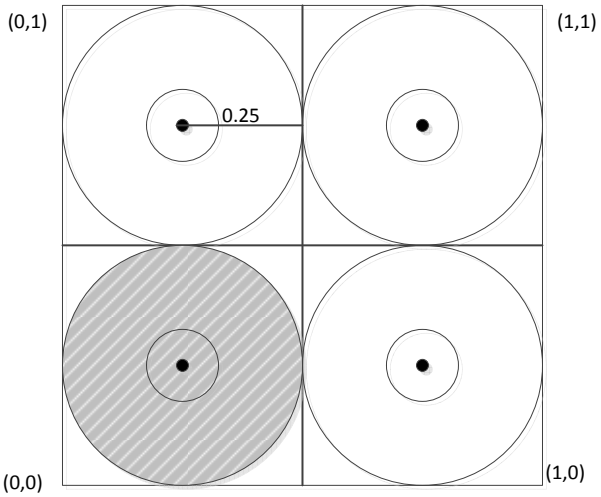

Figure 6: Distortion Demonstration

3. $p_1, p_2, p_3, ...$

$$\begin{aligned}
\sum_{i=n+1}^{\infty} p_i &= \sum_{i=n+1}^{\infty} \frac{9}{10}(\frac{1}{10})^{i-1} \\
&= \frac{9}{10}(\frac{1}{10})^n + \frac{9}{10}(\frac{1}{10})^{n+1} + ... \\
&= \frac{9}{10}(\frac{1}{10})^n[1 + \frac{1}{10} + (\frac{1}{10})^2 + ...] \\
&= \frac{9}{10}\frac{1}{\frac{9}{10}} \\
&= \frac{1}{10}(\frac{1}{10})^{n-1}
\end{aligned}$$

$p_n = \frac{9}{10}(\frac{1}{10})^{n-1}$

Thus $p_n > \sum_{i=n+1}^{\infty} p_i$.

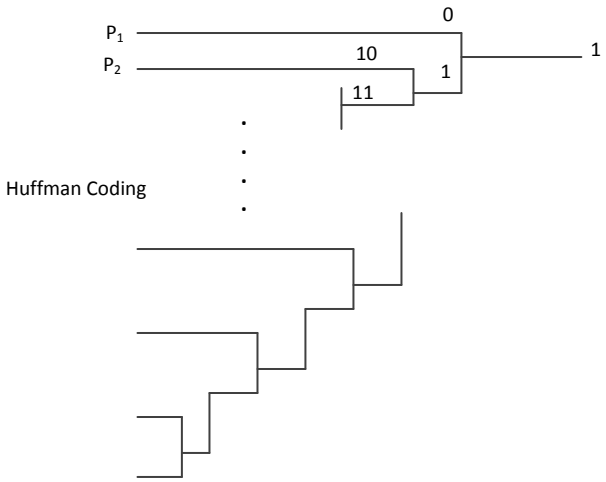

Figure 7: Huffman Coding

So length of the series is 1,2,3,4,.....

4. $\frac{n(n+1)}{2} = 5050 \rightarrow n = 100$
   Length is $100X \lceil \log(100) + 1 \rceil = 800$
   $R = \frac{m(\log(m)+1)}{m(m+1)/2} = \frac{2(\log(m)+1)}{(m+1)}$
5. Covered in the previous course.

EE5585 Data Compression March 5, 2013

Lecture 12

Instructor: Arya Mazumdar *Scribe: Shreshtha Shukla*

1 Review

In the last class we studied the fundamental theorem of source coding, where R(D) is the optimal rate of compression, given the normalized average distortion D. Our task was to find this optimal rate, for which we had the theorem stated as

$$R(D) = \min_{p(\hat{x}|x):\sum p(\hat{x}|x)p(x)d(\hat{x},x)\leq D} I(X,\hat{X})$$

We also proved $R(D) \geq \min I(X,\hat{X})$, known as the converse part of the theorem. The direct part of the theorem known as achievability result is proved using Random Coding method. Also recall, For Bernoulli(p) Random Variables: $R(D) = h(p) - h(D)$ where $h(\cdot)$ is the binary entropy function, when $D = 0, R(D) = h(p)$ that is, the source can be compressed to $h(p)$ bits.

2 Continuous Random Variables:

Suppose we have a source that produce the i.i.d. sequence

$$X_1, X_2, X_3 \dots X_n$$

these can be real or complex numbers.

2.1 1 Bit Representation:

For example: we have gaussian random variable, $X \sim \mathcal{N}(0, \sigma^2)$.

Its pdf is then:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-x^2/2\sigma^2}$$

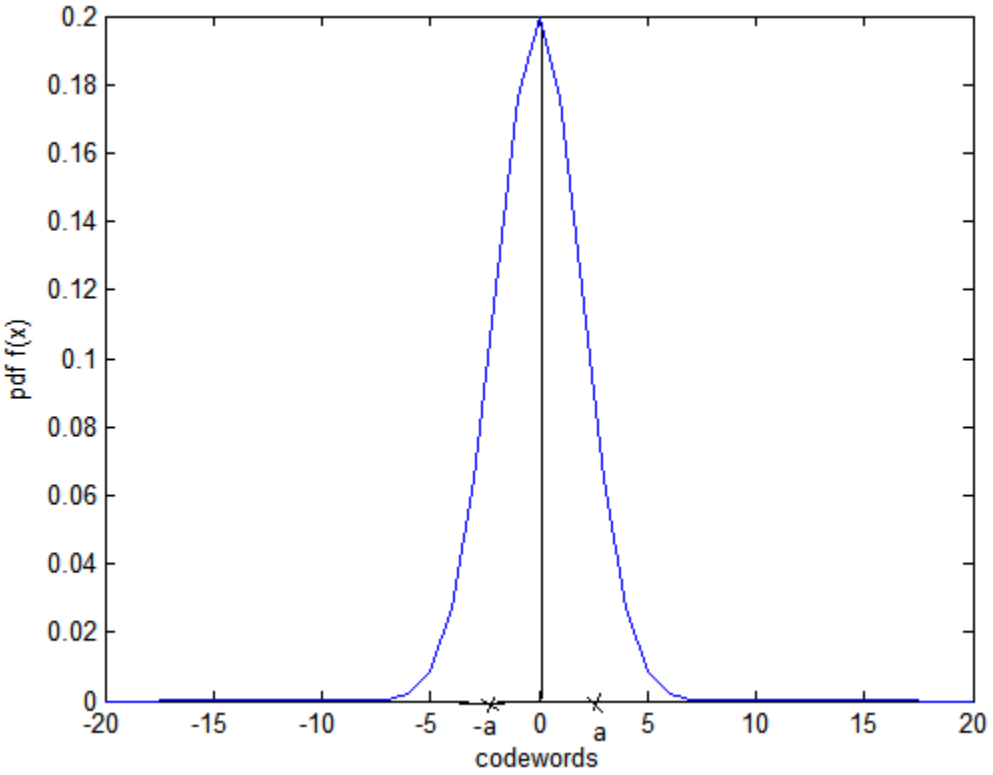

Figure 1: gaussian pdf with 1 bit partition

Now, how can you represent this using 1 bit? With 1 bit, at most we can get the information whether $x \geq 0$ or $x < 0$.

After compressing the file to 1 bit, the next question is: how do we decode it?

For this, some criteria has to be selected, example, we set our codewords based on MSE (mean square error) i.e find the value of 'a' that will minimize the expected squared error

$$\min \; E[(X - \hat{X})^2]$$

$$= \int_{-\infty}^{0} f(x)(x+a)^2 dx \quad + \quad \int_{0}^{\infty} f(x)(x-a)^2 dx$$

$$= \; 2 \int_{0}^{\infty} f(x)(x-a)^2 dx$$

$$= 2\,[\, \int_{0}^{\infty} x^2 f(x) dx \; - \; 2a \int_{0}^{\infty} x f(x) dx \; + \; a^2 \int_{0}^{\infty} f(x) dx \,]$$

$$= (a^2 \; + \; \sigma^2) \; - \; 4a \int_{0}^{\infty} x \, \frac{1}{\sqrt{2\pi\sigma^2}} e^{-x^2/2\sigma^2} dx$$

Let $y = x^2/2\sigma^2$,
$\Rightarrow dy = \frac{xdx}{\sigma^2}$.
So,

$$-4a \int_0^\infty x \frac{1}{\sqrt{2\pi\sigma^2}} e^{-x^2/2\sigma^2} dx \;=\; -2a\sqrt{2\sigma^2/\pi} \int_0^\infty e^{-y} dy$$

$$= -2a\sqrt{2\sigma^2/\pi}$$

hence, $E[(X - \hat{X})^2] \;=\; a^2 \;+\; \sigma^2 \;- 2a\sqrt{2\sigma^2/\pi}$
to minimize this, differentiate with respect to a and set it to 0.

$$\Longrightarrow 2a - 2\sqrt{2\sigma^2/\pi} = 0$$

$$or \quad a = \sigma\sqrt{\frac{2}{\pi}}$$

which is what we choose our codeword.

2.2 2 Bit Representation:

Similarly, for a 2 bit representation,we need to divide the entire region into 4 parts:

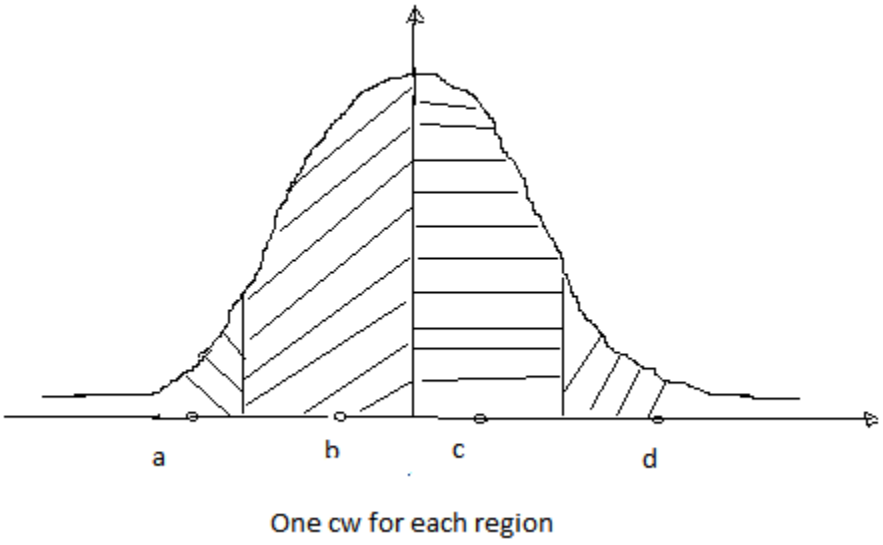

Figure 2: Divided into four parts

Further, it is always good to take a vector instead of a single random variable (i.e a scalar RV). The vector then, lives in an n-dimensional space. In this case also, we need to find the appropriate regions and their associated optimal reconstruction points.

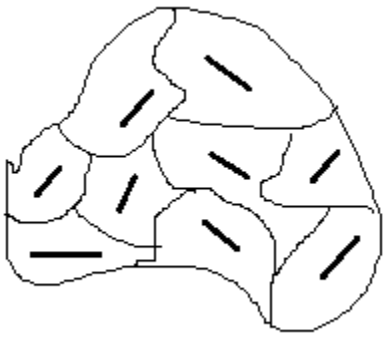

Figure 3: Voronoi Regions in R^nspace

These regions are called the VORONOI Regions and the partitions are known as DIRICHLET'S Partitions.To find the optimal partitions and associated centers, there is an algorithm known as the Llyod's Algorithm.

Briefly the Llyod's algorithm:

We start with some initial set of quantized points, Eg : for 10 bit; $2^{10} = 1024$ points and then find the Voronoi regions for these points. The expected distribution is then optimized. Update the points to those optimal points and again find the Voronoi regions for them. Doing this iteratively converges to optimal values.This algorithm has several names, such as Llyod's Algorithm also known as Expectation Maximization Algorithm. (we know the optimal values using the rate distortion theory so can compare to the convergence result). More detailed study on this will be done in later classes.

3 Entropy for continuous Random Variables

For discrete RVs we have:

$$H(x) = -\sum_{x \in \chi} p(x) \log p(x)$$

similarly for continuous RVs : instead of the pmf we have the pdf $f_X(x)$
Eg, $\mathcal{X} = \mathbf{R}$; $X \sim \mathcal{N}(0, \sigma^2)$
We define Differential Entropy of a continuous random variable as

$$H(X) = -\int_{-\infty}^{\infty} f_X(x) \ln(f_X(x)) dx$$

All the other expressions of conditional entropy, mutual information can be written analogously.
Conditional Entropy: $H(X|Y) = \int_{-\infty}^{\infty} f(x,y) \ln f(x|y) dx dy$.
Mutual Information: $I(X;Y) = H(X) - H(X|Y)$
Also we can have the similar Rate Distortion Theorem for the continous case.

3.1 Examples:

Example 1: Entropy of a gaussian RV:
$X \sim \mathcal{N}(0, \sigma^2)$ $f(x) = 1/\sqrt{2\pi\sigma^2} e^{-x^2/2\sigma^2}$
then

$$H(X) = -\int_{-\infty}^{\infty} 1/\sqrt{2\pi\sigma^2}e^{-x^2/2\sigma^2}\log_e(1/\sqrt{2\pi\sigma^2}e^{-x^2/2\sigma^2})dx$$

$$= -\int_{-\infty}^{\infty} f(x)(\log_e(1/\sqrt{2\pi\sigma^2}) - x^2/2\sigma^2)dx$$

$$= -[\log_e(1/\sqrt{2\pi\sigma^2}) - 1/2\sigma^2\int_{-\infty}^{\infty} x^2 f(x)dx]$$

$$= -[\log_e(1/\sqrt{2\pi\sigma^2}) - 1/2\sigma^2\sigma^2]$$

$$= 1/2(\log_e 2\pi\sigma^2) + 1/2$$

$$= \frac{1}{2}\ln 2\pi e\sigma^2.$$

i.e., $H(X) = \frac{1}{2}\log(2\pi e\sigma^2)$ bits. which is the differential entropy of a gaussian random variable.

3.2 Theorem

CLAIM: A gaussian Random Variable $X \sim \mathcal{N}(0, \sigma^2)$ maximizes the differential entropy among all continuous random variables that have variance of σ^2.

PROOF:

Suppose Z is any random variable with $var(Z) = \sigma^2$ and pdf = g(z),
then $H(X) - H(Z)$ can be written as

$$= -\int f(x)\ln(f(x))dx + \int g(x)\ln(g(x))dx$$

$$= -\int f(x)(\ln(1/\sqrt{2\pi\sigma^2}) - x^2/2\sigma^2)dx + \int g(x)\ln(g(x))dx$$

$$= -\int g(x)\ln(f(x))dx + \int g(x)\ln(g(x))dx$$

$$= \int g(x)\ln(g(x)/f(x))dx$$

$$= D(f||g)$$

$$\geqslant 0$$

Thus $H(X) - H(Z) \geqslant 0$, i.e., H(X)$\geqslant$H(Z).
This method of proof, is called the Maximum Entropy Method.

4 Rate Distortion for gaussian Random Variables:

Suppose we have a gaussian source X : $X_1, X_2X_3...X_n$ are iid.

To compress this data, according to the Rate distortion Theorem:

$$R(D) = \min_{f(\hat{x}|x)} I(X;\hat{X}) \qquad s.t. \int_x \int_{\hat{x}} f(\hat{x}|x)f(x)d(x,\hat{x}) \leq D$$

where $d(\hat{x},x)$ is the euclidean distance. (Proof for this is similar to the discrete case.)

To evaluate, R(D),

Step1: Find a lower bound for $I(X;\hat{X})$

Step2: Find some f(x) that achieves the bound and hence is optimal.

$$I(X;\hat{X}) = H(X) - H(X|\hat{X})$$

now, $H(X/\hat{X}) = H(X-\hat{X}|\hat{X})$

$$= \frac{1}{2}\ln 2\pi e\sigma^2 - H(X-\hat{X}|\hat{X})$$

as conditioning always reduces entropy,

$$\geq \frac{1}{2}\ln 2\pi e\sigma^2 - H(X-\hat{X})$$

and $var(X-\hat{X}) = E[(X-\hat{X})^2]$

$$\geq \frac{1}{2}\ln 2\pi e\sigma^2 - H(N(0, E[(X-\hat{X})^2]))$$

$$\geq \frac{1}{2}\ln 2\pi e\sigma^2 - \frac{1}{2}\ln 2\pi eE[(X-\hat{X})^2]$$

$$\geq \frac{1}{2}\ln(\sigma^2/E[(x-\hat{x})^2]$$

and since we know that $E[(X-\hat{X})^2] \leq D$, always

it implies that for Gaussian,

$$R(D) \geq \frac{1}{2}\ln(\sigma^2/D) \qquad for \quad D \leq \sigma^2$$

$$= 0 \qquad for\ D > \sigma^2$$

After finding a lower bound, we now show that $\exists$ one f(x) for which this bound is achievable (which is the limit of compression for the gaussian RV)i.e we would like to back track and find how the inequalities meet the equality condition.

Suppose we have,

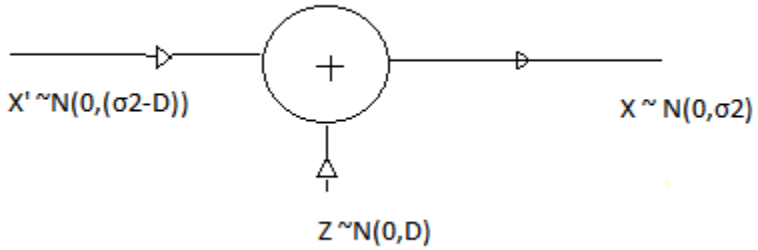

Figure 4: Generating X

where X' is $\hat{X}$

for this case $I(X;\hat{X})= \frac{1}{2}\ln\frac{\sigma^2}{D}$. i.e it achieves the bound.
Hence if there is a Gaussian source producing iid symbols then to encode a vector of length n with resulting quantization error $\leq nD$,
we need at least : $\frac{n}{2}\ln(\sigma^2/D)$ nats or $\frac{n}{2}\log(\sigma^2/D)$ bits to represent it.

EE5585 Data Compression March 8, 2013

Lecture 13

Instructor: Arya Mazumdar *Scribe: Artem Mosesov*

Scalar Quantization

Basics

Being a subset of vector quantization, scalar quantization deals with quantizing a string of symbols (random variables) by addressing one symbol at a time (as opposed to the entire string of symbols.) Although, as one would expect, this is *not* ideal and will not approach any theoretical limits; scalar quantization is a rather simple technique that can be easily implemented in hardware. The simplest form of scalar quantization is uniform quantization.

Given a string, x_1,x_2,...x_n, we first pick a symbol to quantize and disregard the rest. We then quantize this continuous variable to a uniform set of points, as follows:

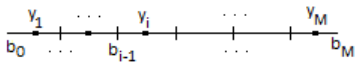

Figure 1: Uniform quantization

So we have M+1 boundaries b_i, and M quantization levels y_i (which fall in the middle of the boundary points). So a continuous number that falls between the boundaries b_{i-1} and b_i gets assigned a quantized value of y_i. Naturally, this introduces signal distortion - an error. The error measure typically used for this is mean squared error (Euclidean distance, as opposed to Hamming distance that's used for binary strings). We call this the quantization error, and recognize that it takes $\log_2(M)$ bits to store the symbol.

Optimization

We note that uniform quantization is only optimal (in the minimum MSE sense) for a uniform distribution. Given an arbitrary PDF (not necessarily uniform), we would like to find an optimal quantization.Let us consider a random variable X with a PDF $f_X(x)$.

The MSE is,

$$\int_{-\infty}^{\infty} (x - Q(x))^2 f_X(x)\, dx$$

where *Q(x)* is the quantized output of X, that is

$$Q(x) = y_i \quad if \quad b_{i-1} \leq x \leq b_i$$

Simplifying the expressions for the error, we have

$$\sigma_q^2 \equiv \text{MSE} = \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x - y_i)^2 f_X(x) dx$$

This, then, becomes an optimization problem - given a maximum distortion rate, we would like to find the optimal location of the quantization points (y_i's and b_i's). Of course, we can always have a very large number of quantization points to keep the distortion low; however, we would like to keep this number low, as to save memory space when storing these values.

Referring back to a uniform distribution, we note that (for a non-uniform pdf), the probability of different y_i's is not the same. That is, at the quantizer output we may see a lot more of a certain quantization point than another. This makes the points a candidate for Huffman coding, as seen earlier in the course. The probability of a particular quantization point is

$$P(Q(x) = y_i) = \int_{b_{i-1}}^{b_i} f_X(x)dx$$

Now we can begin to optimize the average length of the code for the quantization points, which is

$$\sum_{i=1}^{M} l_i \int_{b_{i-1}}^{b_i} f_X(x)dx \quad ,$$

where l_i is the length of the code for y_i. This optimization must occur subject to the following two constraints:

Constraint 1: l_i's satisfy Kraft's inequality.

Constraint 2: $\sigma_q^2 \equiv \text{MSE} = \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x - y_i)^2 f_X(x)dx \leq D$

To see how to simplify this problems, we look again at a uniform quantizer. Lets assume that X (the symbol we want to quantize) is a uniform $\sim U[-L, L]$ variable. The quantization 'lengths' are then $\Delta = \frac{2L}{M}$, as shown in figure 2.

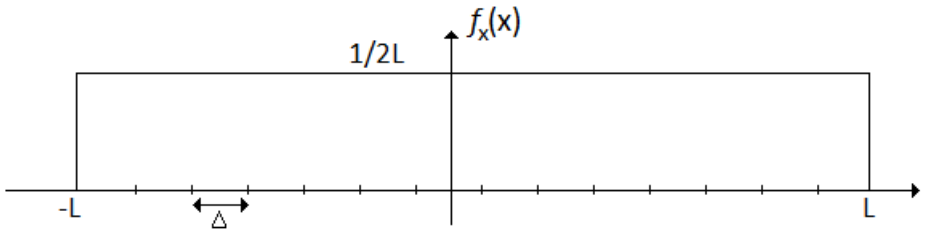

Figure 2: Uniform quantization for uniform random variable

The quantization error then becomes,

$$\sigma_q^2 = \sum_{i=1}^{M} \int_{-L+(i-1)\Delta}^{-L+i\Delta} (x - y_i)\frac{1}{2L}dx$$

The optimal y_i is then $\frac{b_{i-1}+b_i}{2}$. Of course, this is only for a uniform random variable, as initially assumed. We may also notice that the quantization error plot is merely a sawtooth wave with wavelength Δ and amplitude $\frac{\Delta}{2}$. The integral of this is then, $\sigma_q^2 = \frac{\Delta^2}{12}$.

We may think of the quantization error produced by the system as an additive noise - the 'quantization noise'. The power of this noise is then σ_q^2. The idea is shown in Figure 3, below.

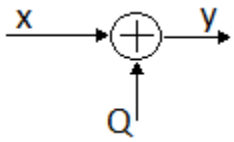

Figure 3: Uniform quantization for uniform random variable

From the figure, we note that the power of the input signal is,

$$\sigma_x^2 = \int_{-L}^{L} x^2 f_X(x) dx = \frac{L^2}{3}$$

Hence, we have, SNR $= 10 \log_{10} (\frac{\sigma_x^2}{\sigma_q^2}) = 20 \log_{10} M$, where M is, as before, the number of quantization levels. Since this is a uniform distribution, Huffman coding will not get us anywhere, and we have the maximum entropy of $20 \log_{10} M$. For an n-bit quantizer then, we get $20 \log_{10} 2^n = 20n \log_{10} 2 \approx 6n$ dB. So the SNR is directly proportional to the number of bits used for quantization - with an increase of one bit correspond to about a 6dB increase of SNR.

Now we take a look at optimum quantization for non-uniform distributions. Similarly, we have

$$\sigma_q^2 = \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x - y_i)^2 f_x(x) dx$$

which we would like to minimize. Often, however, we don't know the exact PDF of the symbols, nor do we know the variance. To overcome this, we use *adaptive quantization*. As we've seen before, one way to do this is to estimate the PDF by observing a string of symbols. This is known as *forward* adaptive quantization.

Going back to minimizing σ_q^2, we want

$$\frac{\delta \sigma_q^2}{\delta y_i} = \frac{\delta}{\delta y_i} \int_{b_{i-1}}^{b_i} (x - y_i)^2 f_x(x) dx =$$

$$= \frac{\delta}{\delta y_i} [\int_{b_{i-1}}^{b_i} x^2 f_x(x) dx - 2y_i \int_{b_{i-1}}^{b_i} x f_x(x) dx + y_i^2 \int_{b_{i-1}}^{b_i} f_x(x) dx] =$$

$$= -2 \int_{b_{i-1}}^{b_i} x f_x(x) dx + 2y_i \int_{b_{i-1}}^{b_i} f_x(x) dx = 0$$

And then we have,

$$y_i = \frac{\int_{b_{i-1}}^{b_i} x f_x(x) dx}{\int_{b_{i-1}}^{b_i} f_x(x) dx} \tag{1}$$

So this is the optimal location of the reconstruction points, given the quantization points. Now we have to find the quantization points. We do this similarly,

$$\frac{\delta \sigma_q^2}{\delta b_i} = 0$$

which gives us the optimal points

$$b_{i-1} = \frac{y_{i-1} + y_i}{2} \tag{2}$$

So what we can do with this is an iterative procedure, where we first initialize the variables, then go back and forth optimizing each one, and (ideally) arriving very close to an optimality point.

Lloyd-Max Algorithm

The *Lloyd-Max* algorithm is an iterative method that does just that. The crude steps (of one of the versions of this algorithm) are as follows:

1. Knowing b_0, assume y_1.
2. Using (1), find b_1.
3. Using (2), find y_2.

and so on...

We also note that since we know the (approximate) signal statistics, we know b_M. Then we have an idea of how much of the error the algorithm made by seeing how close it is to the known value of b_M after the last iteration. If it is too far off, we reinitialize and try again until we are within the accepted tolerance.

Later, we will see a more complex, but better performing method of vector quantization.

EE5585 Data Compression March 12, 2013

Lecture 14

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

Scalar Quantization for Nonuniform Distribution

Suppose we have an input modeled by a random variable X with *pdf* $f_X(x)$ as shown in Figure 1 and we wished to quantize this source using a quantizer with M intervals. The endpoints of the intervals are known as *decision boundaries* and denoted as $\{b_i\}_{i=0}^{M}$, while the representative values $\{y_i\}_{i=1}^{M}$ are called *reconstructive levels*. Then, $Q(X) = y_i$ iff $b_{i-1} < X \leq b_i$, where the quantization operation is defined by $Q(\cdot)$.
The mean squared quantization error (*quantizer distortion*) is given by

$$\sigma_q^2 = E[(X - Q(X))^2] \tag{1}$$
$$= \int_{-\infty}^{\infty} (x - Q(x))^2 f_X(x)dx \tag{2}$$
$$= \int_{b_0}^{b_M} (x - Q(x))^2 f_X(x)dx \tag{3}$$

$$\Rightarrow \sigma_q^2 = \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x - y_i)^2 f_X(x)dx \tag{4}$$

Thus, we can pose the optimal quantizer design problem as the followings:

Given an input *pdf* $f_X(x)$ and the number of quantization levels M in the quantizer, find the decision boundaries $\{b_i\}$ and the reconstruction levels $\{y_i\}$ so as to minimize the mean squared quantization error.

If we know the *pdf* of X, a direct approach to find the $\{b_i\}$ and $\{y_i\}$ that minimize the mean squared quantization error is to set the derivative of (4) with respect to b_j and y_j to zero, respectively. Then,

$$\frac{\partial \sigma_q^2}{\partial y_j} = \frac{\partial}{\partial y_j}[\int_{b_{j-1}}^{b_j} (x - y_j)^2 f_X(x)dx] \tag{5}$$
$$= \frac{\partial}{\partial y_j}[\int_{b_{j-1}}^{b_j} x^2 f_X(x)dx - 2y_j \int_{b_{j-1}}^{b_j} x f_X(x)dx + y_j^2 \int_{b_{j-1}}^{b_j} f_X(x)dx] \tag{6}$$
$$= -2\int_{b_{j-1}}^{b_j} x f_X(x)dx + 2y_j \int_{b_{j-1}}^{b_j} f_X(x)dx = 0 \tag{7}$$

$$\Rightarrow y_j = \frac{\int_{b_{j-1}}^{b_j} x f_X(x)dx}{\int_{b_{j-1}}^{b_j} f_X(x)dx} \tag{8}$$

$$\frac{\partial \sigma_q^2}{\partial b_j} = \frac{\partial}{\partial b_j}[\int_{b_{j-1}}^{b_j} (x - y_j)^2 f_X(x)dx + \int_{b_j}^{b_{j+1}} (x - y_{j+1})^2 f_X(x)dx] \tag{9}$$
$$= (b_j - y_j)^2 f_X(x)dx - (b_j - y_{j+1})^2 f_X(x)dx = 0 \tag{10}$$

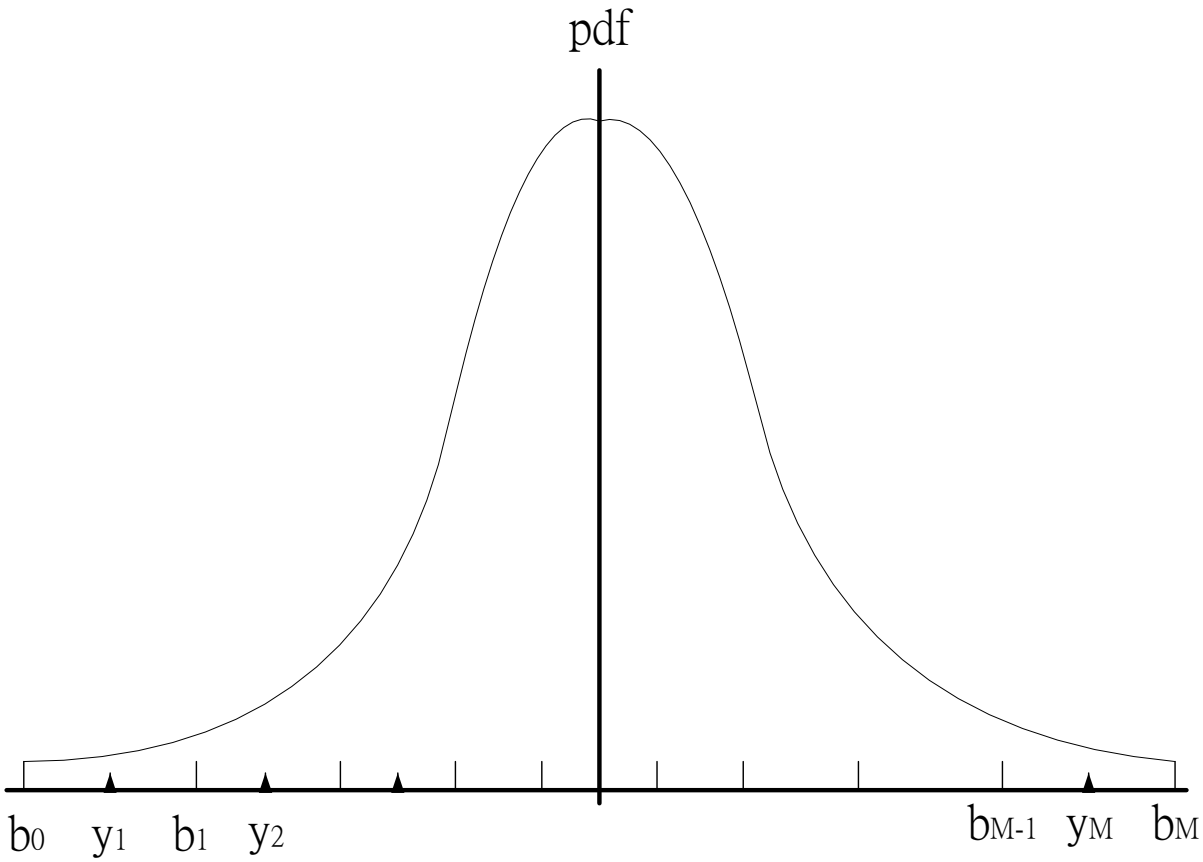

Figure 1: Nonuniform distribution of X.

Then,

$$(b_j - y_j)^2 = (b_j - y_{j+1})^2 \tag{11}$$

$$b_j - y_j = -(b_j - y_{j+1}) \tag{12}$$

$$\Rightarrow b_j = \frac{y_j + y_{j+1}}{2} \tag{13}$$

$$\Rightarrow y_{j+1} = 2b_j - y_j \tag{14}$$

The decision boundary is the midpoint of the two neighboring reconstruction levels. Solving these two equations (8) and (14) listed above will give us the values for the reconstruction levels and decision boundaries that minimize the mean squared quantization error. Unfortunately, to solve for y_j, we need the values of b_j and b_{j-1}, and to solve for b_j, we need the values of y_j and y_{j+1}. Therefore, the Lloyd-Max algorithm is introduced how to solve these two equations (8) and (14) iteratively.

Lloyd-Max Algorithm

Suppose $f_X(x)$ and $b_0 = -\alpha, b_M = +\alpha$ is given, Find $\{b_i\}_{i=0}^M$ and $\{y_i\}_{i=1}^M$. Assume a value for y_1, then
From (8), find b_1
From (14), find y_2
From (8), find b_2
From (14), find y_3
$\vdots$
From (8), find b_{M-1}
From (14), find y_M. Since we know $b_M = +\alpha$, we can directly compute $y'_M = \frac{\int_{b_{M-1}}^{b_M} x f_X(x)dx}{\int_{b_{M-1}}^{b_M} f_X(x)dx}$ and compare it with the previously computed value of y_M. If the difference is less than some tolerance threshold, we can stop. Otherwise, we adjust the estimate of y_1 in the direction indicated by the sign of the difference and repeat the procedure.

Properties of the Optimal Quantizer

The optimal quantizers have a number of interesting properties. We list these properties as follow:
1. Optimal quantizer must satisfy (8) and (14).
2. $EX = EQ(X)$
proof: Since $Q(X) = y_i$ iff $b_{i-1} < X \leq b_i$ and $Pr(Q(X) = y_i) = Pr(b_{i-1} < X \leq b_i)$, then

$$\begin{aligned} EQ(X) &= \sum_{i=1}^{M} y_i Pr(Q(X) = y_i) && (15)\\ &= \sum_{i=1}^{M} y_i Pr(b_{i-1} < X \leq b_i) && (16)\\ &= \sum_{i=1}^{M} \frac{\int_{b_{i-1}}^{b_i} x f_X(x)dx}{\int_{b_{i-1}}^{b_i} f_X(x)dx} \int_{b_{i-1}}^{b_i} f_X(x)dx && (17)\\ &= \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} x f_X(x)dx && (18)\\ &= \int_{b_0}^{b_M} x f_X(x)dx && (19)\\ &= \int_{-\infty}^{+\infty} x f_X(x)dx && (20)\\ &= EX && (21) \end{aligned}$$

The reason of (19) to (20) is that the value of $f_X(x)$ beyond b_0 and b_M is zero.

3. $EQ(X)^2 \leq EX^2$
proof: If $g_X(x) = f_X(x)/(\int_{b_{i-1}}^{b_i} f_X(x)dx)$, then $\int_{b_{i-1}}^{b_i} g_X(x)dx = 1$, $\int_{b_{i-1}}^{b_i} x g_X(x)dx = E_g X$, and $E_g(X - E_g X)^2 \geq 0 \Rightarrow (E_g X)^2 \leq E_g X^2$. Thus,

$$EQ(X)^2 = \sum_{i=1}^{M} y_i^2 Pr(Q(X) = y_i) \quad (22)$$

$$= \sum_{i=1}^{M} \left(\frac{\int_{b_{i-1}}^{b_i} x f_X(x)dx}{\int_{b_{i-1}}^{b_i} f_X(x)dx}\right)^2 \int_{b_{i-1}}^{b_i} f_X(x)dx \tag{23}$$

$$= \sum_{i=1}^{M} \left(\int_{b_{i-1}}^{b_i} x \frac{f_X(x)}{\int_{b_{i-1}}^{b_i} f_X(x)dx} dx\right)^2 \int_{b_{i-1}}^{b_i} f_X(x)dx \tag{24}$$

$$\leq \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} x^2 \frac{f_X(x)}{\int_{b_{i-1}}^{b_i} f_X(x)dx} dx \int_{b_{i-1}}^{b_i} f_X(x)dx \tag{25}$$

$$= \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} x^2 f_X(x)dx \tag{26}$$

$$= \int_{-\infty}^{+\infty} x^2 f_X(x)dx \tag{27}$$

$$= EX^2 \tag{28}$$

4. $\sigma_q^2 = EX^2 - EQ(X)^2$

Lloyd Algorithm

The Lloyd algorith is another method to find $\{b_i\}_{i=0}^M$ and $\{y_i\}_{i=1}^M$. The distribution $f_X(x)$ is assumed known.
Assume $y_1^{(0)}, y_2^{(0)}, \cdots, y_M^{(0)}$ is an initial sequence of reconstruction values $\{y_i\}_{i=1}^M$. Select a threshold ϵ.
1.By Eqn (14). Find $b_0^{(1)}, b_1^{(1)}, \cdots, b_M^{(1)}$.
2.By Eqn (8). Find $y_1^{(1)}, y_2^{(1)}, \cdots, y_M^{(1)}$. And compute $\sigma_q^{2(1)} = \sum_{i=1}^M \int_{b_{i-1}^{(1)}}^{b_i^{(1)}} (x - y_i^{(1)})^2 f_X(x)dx$.
3.By Eqn (14). Find $b_0^{(2)}, b_1^{(2)}, \cdots, b_M^{(2)}$.
4.By Eqn (8). Find $y_1^{(2)}, y_2^{(2)}, \cdots, y_M^{(2)}$. And compute$\sigma_q^{2(2)} = \sum_{i=1}^M \int_{b_{i-1}^{(2)}}^{b_i^{(2)}} (x - y_i^{(2)})^2 f_X(x)dx$.
5.If $|\sigma_q^{2(2)} - \sigma_q^{2(1)}| = \begin{cases} < \epsilon, & \text{then } \textit{stop} \\ \geq \epsilon, & \text{then } \textit{continue the procedure} \end{cases}$

In summary, for each time j, the mean sqaured quantization error $\sigma_q^{2(j)} = \sum_{i=1}^M \int_{b_{i-1}^{(j)}}^{b_i^{(j)}} (x - y_i^{(j)})^2 f_X(x)dx$ is calculated and compare it with previously error value $\sigma_q^{2(j-1)}$. Stop iff $|\sigma_q^{2(j)} - \sigma_q^{2(j-1)}| < \epsilon$; otherwise, continue the same procedure of computing $b_0^{(j+1)}, b_1^{(j+1)}, \cdots, b_M^{(j+1)}$ and $y_1^{(j+1)}, y_2^{(j+1)}, \cdots, y_M^{(j+1)}$ by Eqn (14) and (8) for next time $j+1$.

Vector Quantization

The idea of vector quantization is that encoding sequences of outputs can provide an advantage over the encoding of individual samples. This indicates that a quantization strategy that works with sequences or blocks of outputs would provide some improvement in performance over scalar quantization. Here is an example. Suppose we have two uniform random variables height $X_1 \sim Unif[40, 80]$ and weight $X_2 \sim Unif[40, 240]$ and 3 bits are allowed to represent each random variable. Thus, the weight range is divided into 8 equal intervals and with reconstruction levels $\{52, 77, \cdots, 227\}$; the height range is divided into 8 equal intervals and with reconstruction levels $\{42, 47, \cdots, 77\}$. The two dimensional representation of these two quantizers is shown in Figure 2(a).

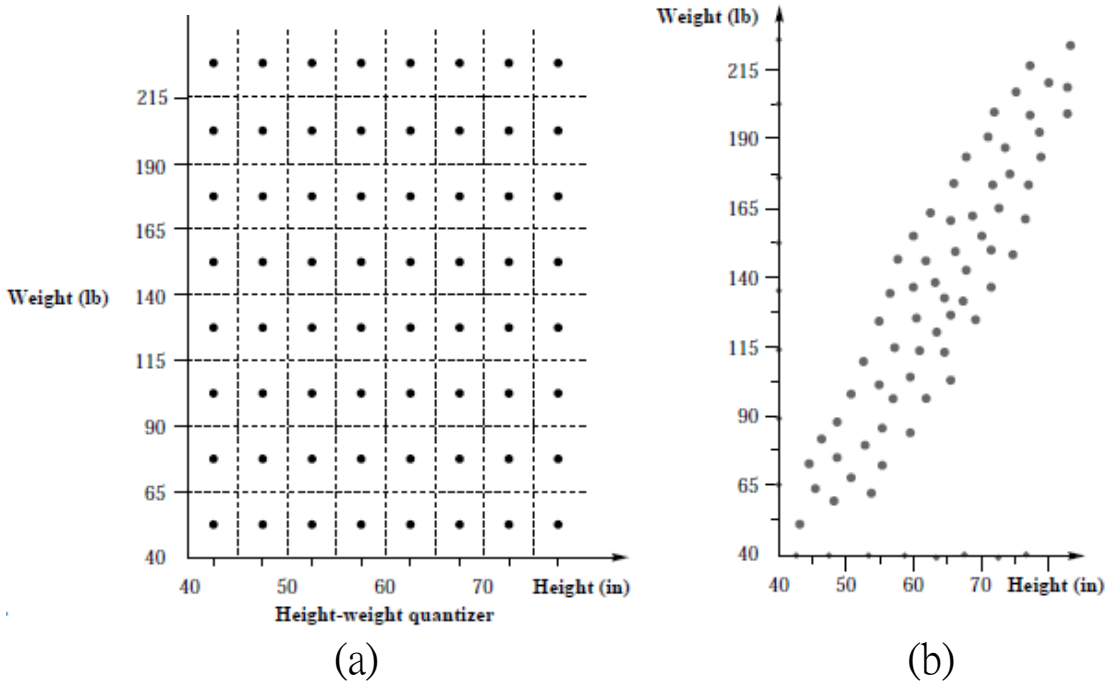

Figure 2: (a)The dimensions of the height/weight scalar quantization. (b)The height-weight vector quantization

However, the height and weight are correlated. For example, a quantizer for a person who is 80 inches tall and weights 40 pounds or who is 42 inches tall and weights 200 pounds is never used. A more sensible approach will use a quantizer like the one shown in Figure 2(b). Using this quantizer, we can no longer quantize the height and weight separately. We will consider them as the coordinates of a point in two dimensions in order to find the closest quantizer output point.

EE5585 Data Compression	March 14, 2013
Lecture 15	
Instructor: Arya Mazumdar	*Scribe: Khem Chapagain*

Scalar Quantization Recap

Quantization is one of the simplest and most general ideas in lossy compression. In many lossy compression applications, we are required to represent each source output using one of a small number of codewords. The number of possible distinct source output values is generally much larger than the number of codewords available to represent them. The process of representing a large - possibly infinite - set of values with a much smaller set is called quantization.

In the previous lectures, we looked at the uniform and nonuniform scalar quantization and moved on to vector quantizers. One thing that's worth mentioning about scalar quanitization is that, for high rate, uniform quantizers are optimal to use. That can be seen from the following derivation. As its name implies, scalar quantizer inputs are scalar values (each input symbol is treated separately in producing the output), and each quantizer output (codeword) represents a single sample of the source output.

- R.V. to quantize: X,
- Set of (decision) boundaries: $\{b_i\}_{i=0}^{M}$,
- Set of reconstruction/quantization levels/points: $\{y_i\}_{i=1}^{M}$, i.e., if x is between b_{i-1} and b_i, then it's assigned the value of y_i.

The Optimal Quantizer has following two properties:

1. $y_i = \dfrac{\int_{b_{i-1}}^{b_i} x\, f_X(x)\, \mathrm{d}x}{\int_{b_{i-1}}^{b_i} f_X(x)\, \mathrm{d}x}$
2. $b_i = \dfrac{y_i + y_{i+1}}{2}$

If we want a high rate quantizer, the number of samples or quantization levels (M) is very large. Consequently, difference between boundaries b_{i-1} and b_i is very small as they are very closely spaced.

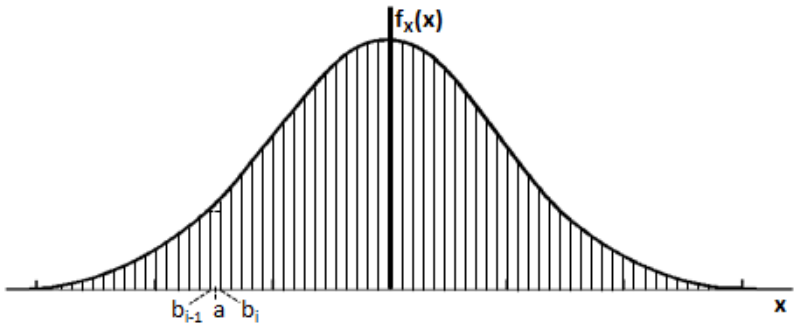

Figure 1: A Gaussian *pdf* with large number of quantization levels (M).

When M is sufficiently large, probability density function (*pdf*), $f_X(x)$, doesn't change much between b_{i-1} and b_i. In that case, the first property above can be written as,

$$\begin{aligned} y_i &\approx \frac{f_X(a) \int_{b_{i-1}}^{b_i} x \, \mathrm{d}x}{f_X(a) \int_{b_{i-1}}^{b_i} \mathrm{d}x}, \quad b_{i-1} \leq a \leq b_i \\ &= \frac{(b_i^2 - b_{i-1}^2)/2}{(b_i - b_{i-1})} \\ &= \frac{b_i + b_{i-1}}{2} \end{aligned}$$

Which says that for very large M, reconstruction level is approximately in the midway between two neighboring boundaries, but we know from the second property above that boundary is exactly between two neighboring reconstruction levels. This means that we have a uniform quantizer since it has all reconstruction levels equally spaced from one another and quantization steps (intervals) are equal. Thus, when we are operating with a large M, it makes sense just to go for the uniform quantizers.

Lloyd Algorithm

In the last class, we talked about Lloyd-Max algorithm to iteratively compute the optimized quantizer levels (y_i's) and boundaries (b_i's). There is another algorithm called Lloyd algorithm, which also iteratively calculates y_i's and b_i's, but in a different way. The Lloyd quantizer that is constructed using an iterative procedure provides the minimum distortion for a given number of reconstruction levels, in other words, it generates the *pdf*-optimized scalar quantizer. The Lloyd algorithm functions as follows (source distribution $f_X(x)$ is assumed to be known):

- Initialization:
  - Assume an initial set of reconstruction values/levels, $\{y_i^{(0)}\}_{i=1}^{M}$, where $^{(0)}$ denotes the 0^{th} iteration.
  - Find decision boundaries, $\{b_i^{(0)}\}_{i=0}^{M}$ from the equation $b_i^{(0)} = \dfrac{y_i^{(0)} + y_{i+1}^{(0)}}{2}$
  - Compute the distortion (variance), $D^{(0)} = \displaystyle\sum_{i=1}^{M} \int_{b_{i-1}^{(0)}}^{b_i^{(0)}} (x - y_i^{(0)})^2 \, f_X(x) \, \mathrm{d}x$
- Update rule: do for $k = 0, 1, 2, ...$
  - $y_i^{(k+1)} = \dfrac{\int_{b_{i-1}^{(k)}}^{b_i^{(k)}} x \, f_X(x) \, \mathrm{d}x}{\int_{b_{i-1}^{(k)}}^{b_i^{(k)}} f_X(x) \, \mathrm{d}x}$, new set of reconstruction levels.
  - $b_i^{(k+1)} = \dfrac{y_i^{(k+1)} + y_{i+1}^{(k+1)}}{2}$, new set of boundaries.

– $D^{(k+1)} = \sum_{i=1}^{M} \int_{b_{i-1}^{(k+1)}}^{b_i^{(k+1)}} (x - y_i^{(k+1)})^2 f_X(x)\,\mathrm{d}x$, new distortion.

- Stop iteration if $|D^{(k+1)} - D^{(k)}| < \epsilon$, where ϵ is a small tolerance > 0; i.e., stop when distortion is not changing much anymore.

We will see later that this algorithm can be generalized to vector quantization.

Vector Quantization

The set of inputs and outputs of a quantizer can be scalars or vectors. If they are scalars, we call the quantizers scalar quantizers. If they are vectors, we call the quantizers vector quantizers. By grouping source outputs together and encoding them as a single block, we can obtain efficient compression algorithms. Many of the lossless compression algorithms take advantage of this fact such as clubbing symbols together or parsing longest phrases. We can do the same with quantization. We will look at the quantization techniques that operate on blocks of data. We can view these blocks as vectors, hence the name "vector quantization." For a given rate (in bits per sample), use of vector quantization results in a lower distortion than when scalar quantization is used at the same rate. If the source output is correlated, vectors of source output values will tend to fall in clusters. By selecting the quantizer output points to lie in these clusters, we have a more accurate representation of the source output.

We have already seen in the prior class that, if we have the data that are correlated, it doesn't make sense to do scalar quantization. We looked at the height(H), weight(W) quantization scheme in two dimensions. We have two random variables (H, W) and will plot the height values along the x-axis and the weight values along the y-axis. In this particular example, the height values are being uniformly quantized to the five different scalar values, as are the weight values. So, we have a total of 25 quantization levels ($M = 25$) denoted by •. The two-dimensional representation of these two quantizers is shown below.

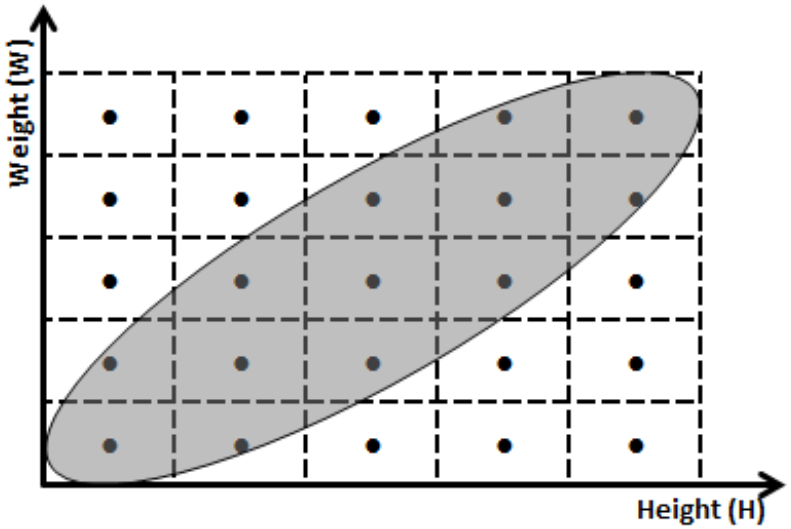

Figure 2: The height, weight scalar quantizers when viewed in two dimensions.

From the figure, we can see that we effectively have a quantizer output for a person who is very tall and weighs minimal, as well as a quantizer output for an individual who is very short but weighs a lot. Obviously, these outputs will never be used, as is the case for many of the other unnatural outputs. A more sensible approach would be to use a quantizer with many reconstruction points inside the shaded region shown in the figure, where we take account of the fact that the height and weight are correlated. This quantizer will have almost all quantization levels (say 23) within this shaded area and no or very few (say 2) quantization levels outside of it. With this approach, the output points are clustered in the area occupied by the most likely inputs. Using this methodology provides a much finer quantization of the input. However, we can no longer quantize the height and weight separately - we have to consider them as the coordinates of a point in two dimensions in order to find the closest quantizer output point.

When data are correlated, we don't have any doubt that quantizing data together (vector quantization) is going to buy us something. It will be shown, when data are independent (no correlation), we still gain from using vector quantization. Scalar quantization always gives something like a rectangular grid similar to what we have seen in the above example. Note that for *pdf*-optimized or optimal quantizer, unlike above example, grids/cells can be of different sizes meaning boundaries don't necessarily have to be uniformly spaced in either dimension. Nevertheless, the fact is that these rectangles don't fill up the space most efficiently. There might be other shapes, like a hexagon or something else, that can fill up the space in a better way with the same number of quantization levels. This idea is captured in vector quantization, where we have this flexibility of using any shape, contrasting the scalar case where only choice is rectangle like shapes.

Exercise: A square and a regular hexagon having equal area.

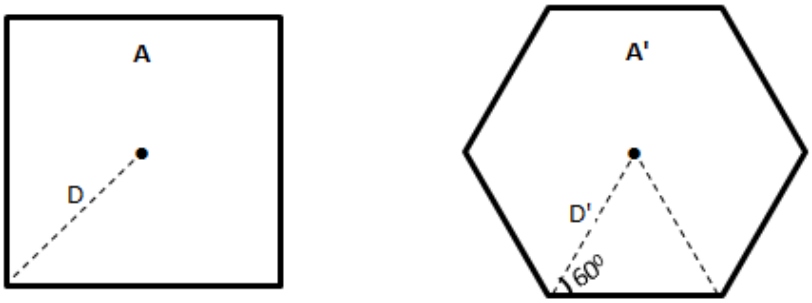

Figure 3: A square and a regular hexagon of same area.

Area of the square,

$$\begin{aligned} A &= (\sqrt{2}D)^2 \\ &= 2D^2 \end{aligned}$$

Area of the hexagon is six times the area of the inside triangle shown,

$$\begin{aligned} A' &= 6\, D' \cos(\frac{\pi}{3})\, D' \sin(\frac{\pi}{3}) \\ &= 6\, \frac{\sqrt{3}}{4} D'^2 \\ &= \frac{3\sqrt{3}}{2} D'^2 \end{aligned}$$

If the areas are equal,

$$\begin{aligned} A' &= A \\ \frac{3\sqrt{3}}{2}D'^2 &= 2D^2 \\ D'^2 &= \frac{4}{3\sqrt{3}}D^2 \\ D'^2 &= \frac{4}{5.196}D^2 \\ D' &= 0.877\,D \\ D' &< D \end{aligned}$$

Even though, both the square and hexagon occupy the same area, the worst case distortion in hexagon (D') is less than the worst case distortion in square (D). Hence, we can intuitively see that hexagon can fill up the same space with less distortion (better packing), which is also known as the shape gain. This is illustrated in the following figure.

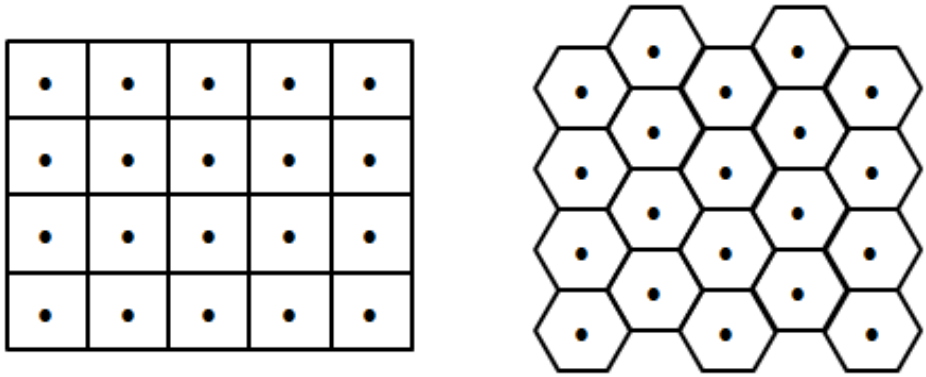

Figure 4: 20 square cells and 20 hexagonal cells packing the equal area.

From earlier discussions, it is clear that for correlated data, vector quantizers are better. We just saw that, even if the the data are not correlated, vector quantizers still offer this shape gain among other advantages. Without having all this intuition and geometry, this fact was captured previously in the rate distortion theory as well.

We can try to extend Lloyd algorithm to vector quantizers. Now, we have a space to quantize instead of a real line. Analogous to scalar case this time we have,

- Given r.v.'s (need not be iid), X_1, X_2, ..., X_n, $\quad X_i \in \mathbb{R}$
- Joint *pdf*, $f_{X_1,...,X_n}(x_1, ..., x_n)$ determines the range (range doesn't always have to be $\mathbb{R}$).
- $\{\underline{Y_i}\}_{i=1}^{M}$, set of quantization levels (n dimensional vectors).
- $\{V_i\}_{i=1}^{M}$, boundaries are hyper-planes or partitions of $\mathbb{R}^n$, i.e., $\bigcup_{i=1}^{M} V_i = \mathbb{R}^n$

Given a set of quantization/reconstruction levels/points, we want find the optimal partitions. Suppose quantization levels are,

$$\{\underline{Y_i}\}_{i=1}^{M}$$

V_i, the Voronoi region of $\underline{Y_i}$, is the region such that any point in V_i is closer to $\underline{Y_i}$ than any other $\underline{Y_j}$.

$$V_i = \{\underline{X} = (x_1, ..., x_n) \in \mathbb{R}^n \mid d(\underline{Y_i}, \underline{X}) < d(\underline{Y_j}, \underline{X}) \; \forall \; j \neq i\}$$

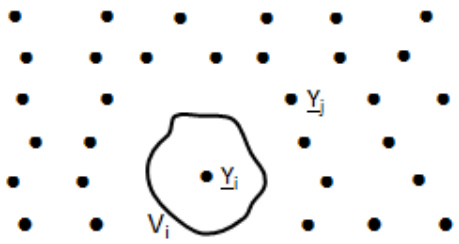

Figure 5: V_i is the Voronoi region of $\underline{Y_i}$.

And the distortion is,

$$\begin{aligned} D &= \int_{\mathbb{R}^n} \|\underline{X} - Q(\underline{X})\|_2^2 \; f_{\underline{X}}(\underline{X}) \; \mathrm{d}\underline{X} \\ &= \sum_{i=1}^{M} \int_{V_i} \|\underline{X} - \underline{Y_i}\|_2^2 \; f_{\underline{X}}(\underline{X}) \; \mathrm{d}\underline{X} \\ &= \sum_{i=1}^{M} \int_{V_i} (\underline{X} - \underline{Y_i})^T (\underline{X} - \underline{Y_i}) \; f_{\underline{X}}(\underline{X}) \; \mathrm{d}\underline{X} \end{aligned}$$

Where $d(.)$ is some distance metric, $Q(.)$ is the quantizer function that maps any value in V_i to $\underline{Y_i}$, and $\|.\|_2^2$ is the Euclidean norm (we can use other norms also).

Linde-Buzo-Gray (LBG) Algorithm

Linde, Buzo, and Gray generalized Lloyd algorithm to the case where the inputs are no longer scalars. It is popularly known as the Linde-Buzo-Gray or LBG algorithm, or the generalized Lloyd algorithm. For the case where the distribution is known, the algorithm looks very much like the Lloyd algorithm described earlier for the scalar case.

- Initialization:
  - Assume an initial set of quantization/reconstruction points/values, $\{\underline{Y_i}^{(0)}\}_{i=1}^{M}$
  - Find quantization/reconstruction/voronoi regions, $V_i^{(0)} = \{\underline{X} \; : \; d(\underline{X}, \underline{Y_i}^{(0)}) < d(\underline{X}, \underline{Y_j}^{(0)}) \; \forall \; j \neq i\}$
  - Compute the distortion, $D^{(0)} = \sum_{i=1}^{M} \int_{V_i^{(0)}} (\underline{X} - \underline{Y_i}^{(0)})^T (\underline{X} - \underline{Y_i}^{(0)}) \; f_{\underline{X}}(\underline{X}) \; \mathrm{d}\underline{X}$
- Update rule: do for $k = 0, 1, 2, ...$
  - $\underline{Y_i}^{(k+1)} = \dfrac{\int_{V_i^{(k)}} \underline{X} \; f_{\underline{X}}(\underline{X}) \; \mathrm{d}\underline{X}}{\int_{V_i^{(k)}} f_{\underline{X}}(\underline{X}) \; \mathrm{d}\underline{X}}$, this represents the point (centroid) that minimizes distortion within that region.

- $V_i^{(k+1)} = \{\underline{X} \ : \ d(\underline{X}, \underline{Y_i}^{(k+1)}) < d(\underline{X}, \underline{Y_j}^{(k+1)}) \ \forall \ j \neq i\}$, new set of reconstruction regions.
- $D^{(k+1)} = \sum_{i=1}^{M} \int_{V_i^{(k+1)}} (\underline{X} - \underline{Y_i}^{(k+1)})^T (\underline{X} - \underline{Y_i}^{(k+1)}) \ f_{\underline{X}}(\underline{X}) \ \mathrm{d}\underline{X}$, updated distortion.

- Stop when $|D^{(k+1)} - D^{(k)}| < \epsilon$, where ϵ is a small tolerance > 0.

This algorithm is not very practical because the integrals required to compute the distortions and centroids are over odd-shaped regions (hyper-space) in n dimensions, where n is the dimension of the input vectors. Generally, these integrals are extremely difficult to compute and trying every vector to decide whether it's in the Voronoi region or not is also unfeasible, making this particular algorithm more of an academic interest. Of more practical interest is the algorithm for the case where we have a training set available. In this case, the algorithm looks very much like the k-means algorithm.

- Initialization:
  - Start with a large set of training vectors, $\underline{X_1}$, $\underline{X_2}$, $\underline{X_3}$, ..., $\underline{X_L}$ from the same source.
  - Assume an initial set of reconstruction values, $\{\underline{Y_i}^{(0)}\}_{i=1}^{M}$
  - Find quantization/voronoi regions, $V_i^{(0)} = \{\underline{X_i} \ : \ d(\underline{X_i}, \underline{Y_i}^{(0)}) < d(\underline{X_i}, \underline{Y_j}^{(0)}) \ \forall \ j \neq i\}$, here we need to partition only L vectors instead of doing it for all vectors.
  - Compute the distortion, $D^{(0)} = \sum_{i=1}^{M} \sum_{\underline{X} \in V_i^{(0)}} (\underline{X} - \underline{Y_i}^{(0)})^T (\underline{X} - \underline{Y_i}^{(0)})$

- Update rule: for $k = 0, 1, 2, ...$
  - $\underline{Y_i}^{(k+1)} = \frac{1}{|V_i^{(k)}|} \sum_{\underline{X} \in V_i^{(k)}} \underline{X}$, find the centroid (center of mass of points), here $|.|$ is the *size of* operator.
  - $V_i^{(k+1)} = \{\underline{X_i} \ : \ d(\underline{X_i}, \underline{Y_i}^{(k+1)}) < d(\underline{X_i}, \underline{Y_j}^{(k+1)}) \ \forall \ j \neq i\}$, new set of decision regions.
  - $D^{(k+1)} = \sum_{i=1}^{M} \sum_{\underline{X} \in V_i^{(k+1)}} (\underline{X} - \underline{Y_i}^{(k+1)})^T (\underline{X} - \underline{Y_i}^{(k+1)})$, new distortion.

- Stop when $|D^{(k+1)} - D^{(k)}| < \epsilon$, where ϵ is a small tolerance > 0.

We didn't know the *pdf* of the data or source, all we had was a set of training data. Yet, when this algorithm stops, we have a set of reconstruction levels ($\underline{Y_i}$'s) and quantization regions (V_i's), which gives us a vector quantizer. This algorithm is a more practical version of the LBG algorithm. Although, this algorithm forms the basis of most vector quantizer designs, there are few issues with this algorithm, for example,

1. **Initializing the LBG Algorithm:** What would be the good set of initial quantization points that will guarantee the convergence? The LBG algorithm guarantees that the distortion from one iteration to the next will not increase. However, there is no guarantee that the procedure will converge to the optimal solution. The solution to which the algorithm converges is heavily dependent on the initial conditions and by picking different subsets of the input as our initial codebook (quantization points), we can generate different vector quantizers.

2. **The Empty Cell Problem:** How do we take care of a situation when one of the reconstruction/quantization regions in some iteration is empty? There might be no points which are closer to a given reconstruction point than any other reconstruction points. This is a problem because in order to update an output point (centroid), we need to take the average value of the input vectors.

Obviously, some strategy is needed to deal with these circumstances. This will be the topic of next class after the Spring Break.

Practical version of LBG algorithm described above is surprisingly similar to the k-means algorithm used in data clustering. Most popular approach to designing vector quantizers is a clustering procedure known as the k-means algorithm, which was developed for pattern recognition applications. The k-means algorithm functions as follows: Given a large set of output vectors from the source, known as the training set, and an initial set of k representative patterns, assign each element of the training set to the closest representative pattern. After an element is assigned, the representative pattern is updated by computing the centroid of the training set vectors assigned to it. When the assignment process is complete, we will have k groups of vectors clustered around each of the output points (codewords).

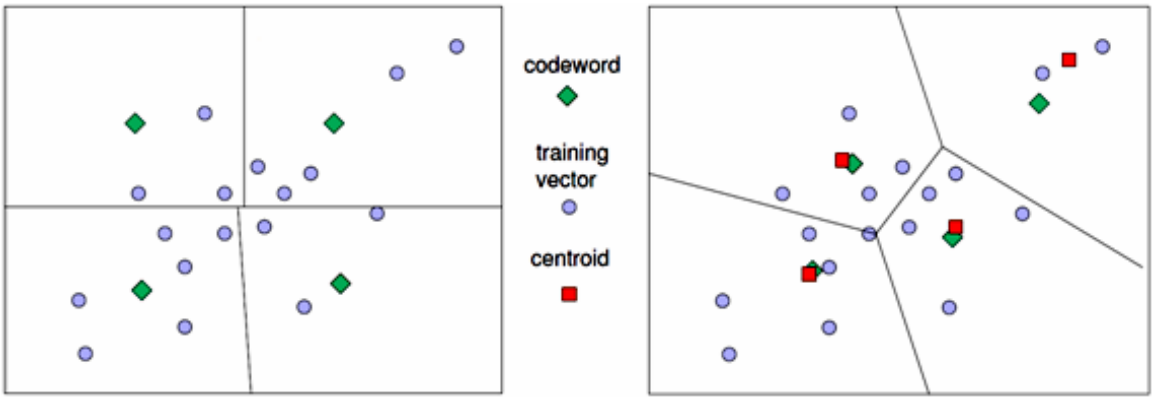

Figure 6: Initial state (left) and final state (right) of a vector quantizer.

We have seen briefly how we can make use of the structure exhibited by groups, or vectors, of values to obtain compression. Since there are different kinds of structure in different kinds of data, there are a number of different ways to design vector quantizers. Because data from many sources, when viewed as vectors, tend to form clusters, we can design quantizers that essentially consist of representations of these clusters.

EE5585 Data Compression March 26, 2013

Lecture 16

Instructor: Arya Mazumdar *Scribe: Fangying Zhang*

1 Review of Homework 6

The review is omitted from this note.

2 Linde-Buzo-Gray (LBG) Algorithm

Let us start with an example of height/weight data.

H (in)	W (lb)
65	170
70	170
56	130
80	203
50	153
76	169

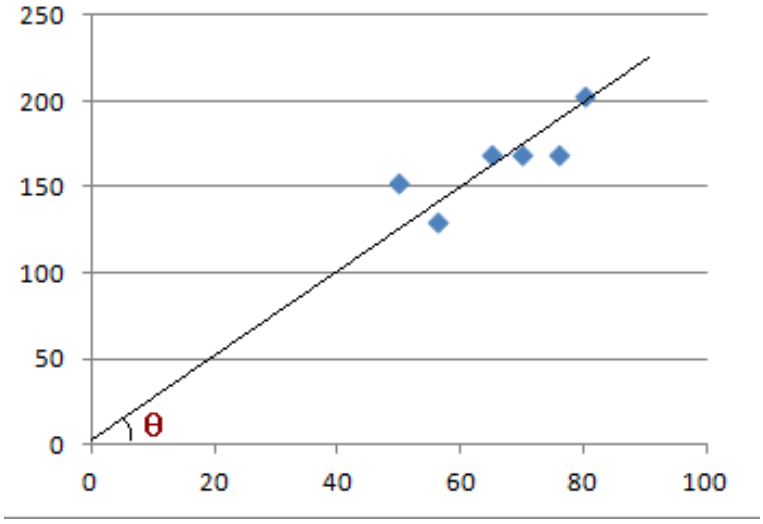

Figure 1:

We need to find a line such that most points are around this line. This means the distances from the points to the line is very small. Suppose $W = 2.5H$ is the equation of the straight line. Let A be a matrix to project values to this line and its perpendicular direction.

We have

$$A = \begin{bmatrix} 0.37 & 0.92 \\ -0.92 & 0.37 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$X = \begin{bmatrix} 65 \\ 170 \end{bmatrix}, \begin{bmatrix} 70 \\ 170 \end{bmatrix}, \begin{bmatrix} 56 \\ 130 \end{bmatrix} \cdots$$

Then by rotating x-axis, we have a new axis:

$$Y = AX$$

From the above equation, we can get a set of new H-W table as following:

H (new)	W (new)
182	3
184	-2
191	-4
218	1
161	10
181	-9

We can get back original data by rotating the axis again: $X = A^{-1}Y$, for example,

$$\begin{bmatrix} 65 \\ 170 \end{bmatrix} = A^{-1} \begin{bmatrix} 182 \\ 3 \end{bmatrix}$$

where,,

$$A^{-1} = \begin{bmatrix} 0.37 & -0.92 \\ 0.92 & 0.37 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} = A^T.$$

Note,

$$\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} * \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Now, neglect the right column of the new table and set them to 0, that is:

H (new)	W (new)
182	0
184	0
191	0
218	0
161	0
181	0

Multiplying each row by A^T we have,

H	W
68	169
68	171
53	131
81	201
60	151
67	168

Compare this table with the original table. We find that the variations are not large. So this new table can be used.

3 Transform Coding

1. For an orthonormal transformation matrix A,

$$A^T = A^{-1} \Leftrightarrow A^T A = AA^T = I. \tag{1}$$

Suppose $y = Ax, \hat{x} = A^{-1}\hat{y}$, where $\hat{y}$ is the stored/compressed value. We introduce error

$$\begin{aligned} &= \| y - \hat{y} \|_2^2 && (2)\\ &= \| Ax - A\hat{x} \|_2^2 && (3)\\ &= \| A(x - \hat{x}) \|_2^2 && (4)\\ &= [A(x - \hat{x}]^T * [Ax - \hat{x}] && (5)\\ &= (x - \hat{x})^T * A^T * A(x - \hat{x}) && (6)\\ &= (x - \hat{x})^T * (x - \hat{x}) && (7)\\ &= \| x - \hat{x} \|_2^2 && (8) \end{aligned}$$

If we do not introduce a lot of errors in y, then there won't be a lot of error for x.

2.

$$E[y^T y] = E[x^T ATAx] = E[x^T x]. \tag{9}$$

which means the input and output energies are the same. This is known as Parseval's identity.

4 Hadamard Transform

Suppose,

$$A = 1/\sqrt{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{10}$$

-1 is shaded as black in the figure below.:

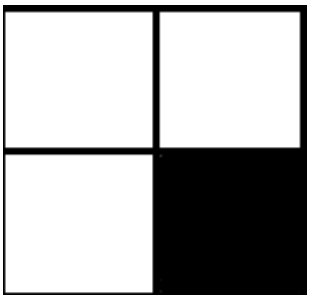

Figure 2:

Use Kronecher Product to define: $A_4 = A_2 \oplus A_2$ we have

$$A_4 = 1/\sqrt{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

The Kronecker product is defined to be,

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \otimes \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

$$= \begin{bmatrix} b_{11}M & b_{12}M \\ b_{21}M & b_{22}M \end{bmatrix}$$

where,

$$M = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}.$$

A_4 can be represented as below.

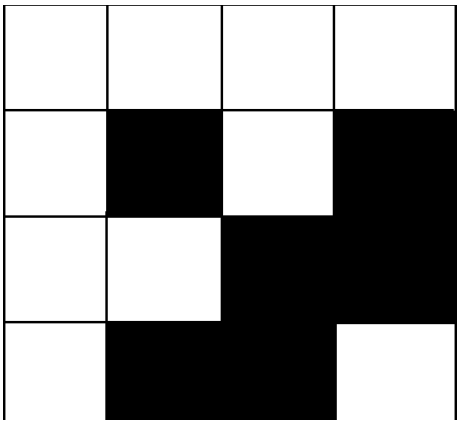

Figure 3:

Then we can have $A_{16} = A_4 \oplus A_4$ in this similar way.

5 Open Problem: Fix-free code, Conjectures, Update-efficiency

Instructor's note: This section is removed (the reason can be explained in person). Please consult your notes.

EE5585 Data Compression March 28, 2013

Lecture 17

Instructor: Arya Mazumdar *Scribe: Chendong Yang*

Solution to Assignment 2

Problem 4 Let X be zero mean and σ^2 variance Gaussian random variable. That is

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

and let the distortion measure be squared error. Find the optimum reproduction points for 1-bit scalar quantization and the expected distortion for 1-bit quantization. Compare this with the rate-distortion function of a Gaussian random variable.

Solution Let x be the source and $\hat{x}$ be quantizer output. Our goal is to find optimal reproduction points -a and a (**figure 1**) such that the distortion is minimize in terms of MSE. i.e. minimize $D = E[(x - \hat{x})^2]$.

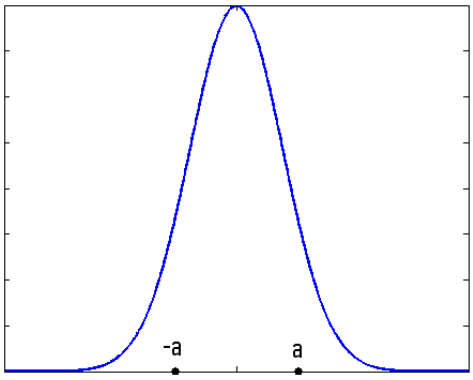

Figure 1: PDF of problem 4 with quantization level **a** and **-a**

$$\begin{aligned}
D = E[(x-\hat{x})^2] &= \int_{-\infty}^{0} f_X(x)(x+a)^2 dx + \int_{0}^{\infty} f_X(x)(x-a)^2 dx \\
&= 2\int_{0}^{\infty} f_X(x)(x-a)^2 dx \\
&= 2\left[\int_{0}^{\infty} a^2 f_X(x)dx + \int_{0}^{\infty} x^2 f_X(x)dx - 2a\int_{0}^{\infty} x f_X(x)dx\right] \\
&= a^2 + \sigma^2 - \frac{4a\sigma^2}{\sqrt{2\pi\sigma^2}} \int_{0}^{\infty} e^{-y} dy \quad (\text{Let } y = -\frac{x^2}{2\sigma^2}) \\
&= a^2 + \sigma^2 - \frac{4a\sigma^2}{\sqrt{2\pi\sigma^2}}
\end{aligned}$$

Now we take partial differentiation of $E[(x - \hat{x})^2]$ with respect to a and set it to zero,

$$\frac{\partial E[(x-\hat{x})^2]}{\partial a} = 2a - \frac{4\sigma^2}{\sqrt{2\pi\sigma^2}} = 0$$
$$\Rightarrow a^* = \sigma\sqrt{\frac{2}{\pi}}$$

So **two optimum reproduction points are** $a^* = \sigma\sqrt{\frac{2}{\pi}}$ and $-a^* = -\sigma\sqrt{\frac{2}{\pi}}$. Substitute $a^* = \sigma\sqrt{\frac{2}{\pi}}$ back to expectd distortion expression we just found, and the **expected distortion**

$$D = E[(x-\hat{x})^2] = \sigma^2\frac{2}{\pi} + \sigma^2 - 4\sigma^2\frac{1}{\sqrt{2\pi}}\sqrt{\frac{2}{\pi}}$$
$$= \frac{\pi-2}{\pi}\sigma^2$$

We know that binary rate-distortion function of a Gaussian random variable is

$$R(D) = \frac{1}{2}\log_2\frac{\sigma^2}{D}$$
$$\Rightarrow D_{opt} = \frac{\sigma^2}{2^{2R(D)}}$$

In our case rate $R = 1$, so that $D_{opt} = \frac{\sigma^2}{4} < \frac{\pi-2}{\pi}\sigma^2$, **which means the distortion rate bound is smaller than the distortion in our case**.

Problem 6 Let $\{X\}_{i=o}^{\infty}$ be an i.i.d binary sequence with probability of 1 being 0.3. Calculate $F(01110) = P(0.X_1X_2X_3X_4X_5 < 0.01110)$. How many bits of $F = 0.X_1X_2X_3X_4X_5....$ can be known for sure if it is not known how the sequence 0.01110... continues?

Solution

$$\begin{aligned} F(01110) &= Pr(0.X_1X_2X_3X_4X_5 < 0.01110) \\ &= Pr(X_1 = 0, X_2 < 1) + Pr(X_1 = 0, X_2 = 1, X_3 < 1) + Pr(X_1 = 0, X_2 = 1, X_3 = 1, X_4 < 1) \\ &= 0.7^2 + 0.7^2 \cdot 0.3 + 0.7^2 \cdot 0.3^2 \\ &= 0.6811 \end{aligned}$$

From the source, we have only observed the first five bits, 01110 ,which can be continued with an arbitary sequence. However for an arbitary sequence that starts with the sequence 01110 we have

$$F(01110000\ldots) \leq F(01110X_6X_7X_8\ldots) \leq F(01110111\ldots).$$

We know that

$$F(01110000\ldots) = F(01110) = (0.6811)_{10} = (0.101011100101...)_2$$

However, we also know that $F(01110111\ldots) = F(01111)$.

$$\begin{aligned} F(01111) &= Pr(0.X_1X_2X_3X_4X_5 < 0.01111) \\ &= 0.7^2 + 0.7^2 \cdot 0.3 + 0.7^2 \cdot 0.3^2 + 0.7^2 \cdot 0.3^3 \\ &= (0.69433)_{10} = (0.101100011011...)_2 \end{aligned}$$

So comparing binary representation of $F(01110)$ and $F(01111)$ we observe that **we are sure about 3 bits: 101**.

Problem 3 Consider the following compression scheme for binary sequences. We divide the binary sequences into blocks of size 16. For each block if the number of zeros is greater than or equal to 8 we store a 0, otherwise we store a 1. If the sequence is random with probability of zero 0.9, compute ther rate and average distortion (Hamming metric). Compare your result with the corresponding value of rate distortion function for binary sources.

Solution It is easy to see that $\textbf{rate} = \mathbf{R(D)} = \frac{\mathbf{1}}{\mathbf{16}}$. We can find the optimum distortion D_{opt} when the rate is $\frac{1}{16}$ according to binary rate-distortion function,

$$R(D) = h(p) - h(D_{opt})$$

where h(.) is binary entropy function and p is the source probability of zero.

$$\begin{aligned} D_{opt} &= h^{-1}\left[h(p) - R(D)\right] \\ \Rightarrow D_{opt} &= h^{-1}\left[h(0.1) - \frac{1}{16}\right] \\ \Rightarrow D_{opt} &\approx 0.08 \end{aligned}$$

In our source compression scheme, the distortion is summarized in Table 1. Assume that two quantization levels are 000... and 111... respectively

Number of 0's in block	Encoding	Probability	Distortion
0	1	$\binom{16}{0}(0.1)^{16}$	0
1	1	$\binom{16}{1}(0.1)^{15}\cdot(0.9)^{1}$	1
2	1	$\binom{16}{2}(0.1)^{14}\cdot(0.9)^{2}$	2
3	1	$\binom{16}{3}(0.1)^{13}\cdot(0.9)^{3}$	3
4	1	$\binom{16}{4}(0.1)^{12}\cdot(0.9)^{4}$	4
5	1	$\binom{16}{5}(0.1)^{11}\cdot(0.9)^{5}$	5
6	1	$\binom{16}{6}(0.1)^{10}\cdot(0.9)^{6}$	6
7	1	$\binom{16}{7}(0.1)^{9}\cdot(0.9)^{7}$	7
8	0	$\binom{16}{8}(0.1)^{8}\cdot(0.9)^{8}$	8
9	0	$\binom{16}{9}(0.1)^{7}\cdot(0.9)^{9}$	7
10	0	$\binom{16}{10}(0.1)^{6}\cdot(0.9)^{10}$	6
11	0	$\binom{16}{11}(0.1)^{5}\cdot(0.9)^{11}$	5
12	0	$\binom{16}{12}(0.1)^{4}\cdot(0.9)^{12}$	4
13	0	$\binom{16}{13}(0.1)^{3}\cdot(0.9)^{13}$	3
14	0	$\binom{16}{14}(0.1)^{2}\cdot(0.9)^{14}$	2
15	0	$\binom{16}{15}(0.1)^{1}\cdot(0.9)^{15}$	1
16	0	$\binom{16}{16}(0.9)^{16}$	0

Table 1: Distortion Summary

Refer to the table we can compute the **average distortion**,

$$D_{ave} = E[d(x,\hat{x})] = \sum_{i=0}^{16} Pr(\text{Number of 0's} = i)\cdot d(x_i,\hat{x}_i) = 1.6$$

So the normalized distortion $D_{norm} = \frac{D_{ave}}{16} = 0.1$, **which is larger than D_{opt} we found before.**

Problem 1 Design a 3-bit uniform quantizer(by specifying the decision boundaries and reconstruction levels) for a source with following pdf:

$$f_X(x) = \frac{1}{6} e^{-\frac{|x|}{3}}$$

Solution Let Δ be decision boundary and function $Q(.)$ be uniform quantizer. Since it is a 3-bit uniform quantizer, we need $2^3 = 8$ reconstruction levels as following (Note we only consider positive region here because the distribution is symmetric):

$$Q(x) = \begin{cases} \frac{\Delta}{2} & 0 \leq x \leq \Delta \\ \frac{3\Delta}{2} & \Delta < x \leq 2\Delta \\ \frac{5\Delta}{2} & 2\Delta < x \leq 3\Delta \\ \frac{7\Delta}{2} & 3\Delta < x < \infty \end{cases}$$

Therefore, the expression of Mean Squared Quantization Error becomes

$$\begin{aligned} \sigma_q^2 = E\left[(X - Q(x))^2\right] &= \int_{-\infty}^{\infty} (x - Q(x))^2 f_X(x) dx \\ &= 2\sum_{i=1}^{i=3} \int_{(i-1)\Delta}^{i\Delta} \left(x - \frac{2i-1}{2}\Delta\right)^2 f_X(x) dx + 2\int_{3\Delta}^{\infty} \left(x - \frac{7\Delta}{2}\right)^2 f_X(x) dx \end{aligned}$$

To find the optimal value of Δ. we simply take a derivative of this equation and set it equal to zero,

$$\frac{\partial \sigma_q^2}{\partial \Delta} = -\sum_{i=1}^{i=3} (2i-1) \int_{(i-1)\Delta}^{i\Delta} \left(x - \frac{2i-1}{2}\Delta\right) f_X(x) dx - 7\int_{3\Delta}^{\infty} \left(x - \frac{7\Delta}{2}\right)^2 f_X(x) dx = 0$$

Substitute $f_X(x) = \frac{1}{6} e^{-\frac{|x|}{3}}$ into above equation. After some calculus and algebra, we get

$$\boldsymbol{\Delta \approx 3.101}$$

The optimuml 3-bit uniform quantizer shows in **figure 2**

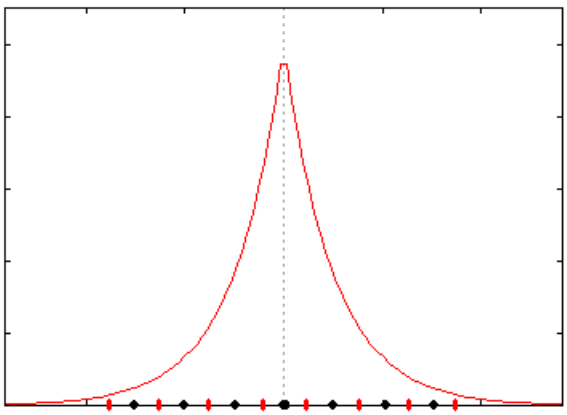

Figure 2: Black dots represent optimuml quantization levels spaced by $\Delta \approx 0.3101$. Red dots represent optimum reconstruction levels, which are set in the middle of two adjacent quantization levels

Problem 2 What is the mean and variance of the random variable of Problem 1 above? Derive

the mean of the output of uniform quantizer you designed for the above problem. What is the mean of the optimum quantizer for this distribution?

Solution

$$\textbf{Mean:} \quad E[X] = \int_{-\infty}^{\infty} x f_X(x) dx = \int_{-\infty}^{\infty} x \cdot \frac{1}{6} e^{-\frac{|x|}{3}} dx = 0$$

$$\textbf{Variance:} \quad E[X^2] = \int_{-\infty}^{\infty} x^2 f_X(x) dx = 2 \int_{0}^{\infty} x^2 \cdot \frac{1}{6} e^{-\frac{x}{3}} dx = 18$$

$$\textbf{Optimum Quantizer Mean:} \quad E[Q(x)] = E[X] = 0$$

The **mean of the output of designed uniform quantizer** is

$$E[Q(x)] = \sum_{i=-4}^{i=4} Pr\left(Q(x) = \frac{2i-1}{2}\Delta\right)\left(\frac{2i-1}{2}\Delta\right)$$
$$= 0 \quad \text{(since it is a odd function)}$$

where $Pr\left(Q(x) = \frac{2i-1}{2}\Delta\right) = \int_{(i-1)\Delta}^{i\Delta} f_X(x) dx$

Problem 5 Consider a source X uniformly distributed on the set {1,2,...m}. Find the rate distortion function for this source with Hamming distance; that is, $\mathrm{d}(x, \hat{x}) = 0$ when $x = \hat{x}$ and $\mathrm{d}(x, \hat{x}) = 1$ when $x \neq \hat{x}$.

Solution Solution will be given in the next class.

1 Kolmogorov Complexity

Let us briefly talk about the concept of *Kolmogorov complexity*. First, we should ask ourself "what is a computer?". Generally the computer is a *Finite State Machine(FSM)*, consisting of read tape, output tape and work tape. (**figure 3**). This is called a Turing machine (Alan Turing).

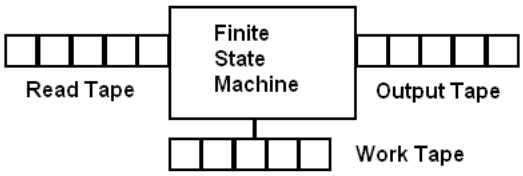

Figure 3: Illustration of a simple Finite State Machine

A Turing machine is able to simulate any other computer (*Church-Turing Thesis*). Any real-world computation can be translated into an equivalent computation involving a Turing machine.

A sequence 010101.... ("01" repeat 10000 times) can be translated to the sentence "Repeat "01" 10000 times". Another example, a sequence 141592... can be translated to the sentence "First 6 digits after the point in π."

Now let us define Kolmogorov complexity. For any sequence x

$$K_U(x) = \min_{p:U(p)=x} l(p)$$

where p is a computer program that computes x and halts, U is a computer.

Instructor note: Incomplete. Please consult your notes.

EE5585 Data Compression April 02, 2013

Lecture 18

Instructor: Arya Mazumdar *Scribe: Shashanka Ubaru*

Solutions for problems 1 - 4 and 6 of HW2 were provided in the previous lecture and also the concepts of Turing machines and Kolmogorov Complexity were introduced . In this lecture,

1. Solution for the fifth problem of HW2 is provided.
2. Kolmogorov Complexity is defined.
3. Properties (Theorems related to) Kolmogorov Complexity are stated and proved.
4. Concept of Incompressible Sequences is introduced.

(Reference for these topics : Chapter 14, "Elements of Information Theory" 2nd ed - T. Cover, J. Thomas (Wiley, 2006).)

Solution for Problem 5 of Homework 2

Given: A source X uniformly distributed on the set $\{1, 2, \ldots\ldots, m\}$. That is, $\Pr(x = i) = \frac{1}{m}$.

$R(D) =?$ with Hamming distortion,

$$d(x, \widehat{x}) = \begin{cases} 0 & if\ x = \widehat{x} \\ 1 & if\ x \neq \widehat{x} \end{cases}$$

We know that the rate distortion is given by,

$$R(D) = \min_{p(\widehat{x}|x): E\{d(x,\widehat{x}) \leq D\}} I(X; \widehat{X})$$

This optimization equation seems difficult to solve. So, a good trick is to find a lower bound for $I(X; \widehat{X})$ subjected to the constraint mentioned and come up with an example that achieves this lower bound given the constraint on $p(\widehat{x} \mid x)$. (Recall : This technique was used to find $R(D)$ for Binary and Gaussian random variables also)

By definition,

$$\begin{aligned} I(X; \widehat{X}) &= H(X) - H(X \mid \widehat{X}) \\ &= \log m - H(X \mid \widehat{X}) \end{aligned}$$

For binary random variable, we had equaled $H(X \mid \widehat{X})$ to $H(X - widehatX \mid \widehat{X})$, but here this is not true. So, we define a new random variable Y,

$$Y = \begin{cases} 0 & if\ X = \widehat{X} \\ 1 & if\ X \neq \widehat{X} \end{cases}$$

$H(X \mid \widehat{X})$ is the uncertainity in X if $\widehat{X}$ is known and we have,

$$\begin{aligned} H(X \mid \widehat{X}) &\leq H(X, Y \mid \widehat{X}) \\ &= H(Y \mid \widehat{X}) + H(X \mid \widehat{X}, Y). \end{aligned}$$

Substituting,

$$\begin{aligned} I(X;\widehat{X}) &\geq H(X) - H(Y \mid \widehat{X}) - H(X \mid \widehat{X}, Y) \\ &\geq \log m - H(Y) - H(X \mid \widehat{X}, Y) \end{aligned}$$

as $H(Y) \geq H(Y \mid \widehat{X})$. Now consider $H(X \mid \widehat{X}, Y)$,

$$H(X \mid \widehat{X}, Y) = \Pr(Y=0) H(X \mid \widehat{X}, Y=0) + \Pr(Y=1) H(X \mid \widehat{X}, Y=1)$$

If $Y = 0 \Rightarrow X = \widehat{X} \Rightarrow H(X \mid \widehat{X}, Y=0)$ there is no uncertainity and for a given $\widehat{X}$, there are only M-1 choices for X.

$$\begin{aligned} H(X \mid \widehat{X}, Y) &= \Pr(Y=1) \log(M-1) \\ &= \Pr(X \neq \widehat{X}) \log(M-1) \end{aligned}$$

and $H(Y) = h\left(\Pr(X \neq \widehat{X})\right)$ then,

$$\begin{aligned} I(X;\widehat{X}) &\geq \log m - h\left(\Pr(X \neq \widehat{X})\right) - \Pr(X \neq \widehat{X}) \log(M-1) \\ Ed(x;\widehat{x}) &= 1.\Pr(X \neq \widehat{X}) + 0.\Pr(X = \widehat{X}) \\ &= \Pr(X \neq \widehat{X}) \leq D \\ I(X;\widehat{X}) &\geq \log m - \underbrace{D \log(M-1) - h(D)}_{both\ are\ increasing\ functions} \end{aligned}$$

Example to show that, this lower bound is achieved

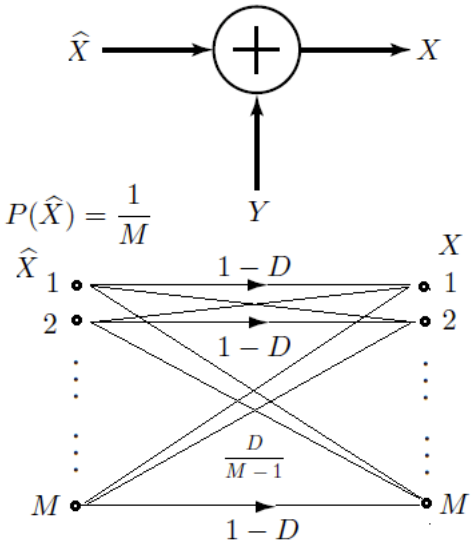

Figure 1: System that achieves the lower bound

Consider the system shown in Figure 1. $\widehat{X} \in \{1, 2, \ldots\ldots, M\}$ with $\Pr(\widehat{X}) = \frac{1}{M}$. If the distortion is D then, $\Pr(X = i \mid \widehat{X} = i) = 1 - D$ and $\Pr(X = i \mid \widehat{X} = j) = \frac{D}{M-1}$ $\forall i \neq j$ as shown in the figure. The probability X is,

$$\begin{aligned}
\Pr(X=i) &= \Pr(\widehat{X}=i)\Pr(X=i \mid \widehat{X}=i) + \sum_{i\neq j}\Pr(\widehat{X}=j)\Pr(X=i \mid \widehat{X}=j) \\
&= \frac{1}{M}(1-D) + \sum_{i\neq j}\frac{1}{M}\frac{D}{M-1} \\
&= \frac{1-D}{M} + \frac{D}{M} \\
&= \frac{1}{M}
\end{aligned}$$

So, X are equiprobable. The mutual information is given by,

$$\begin{aligned}
I(X;\widehat{X}) &= \log m - H(X \mid \widehat{X}) \\
H(X \mid \widehat{X}) &= -\sum_{i=1}^{M}\Pr(\widehat{X}=i)H(X \mid \widehat{X}=i) \\
&= \frac{1}{M}\sum_{i=1}^{M}H(X \mid \widehat{X}=i) \\
&= H(X \mid \widehat{X}=i)
\end{aligned}$$

We have, $\Pr(X=i \mid \widehat{X}=i) = 1-D$ and $\Pr(X=i \mid \widehat{X}=j) = \frac{D}{M-1}\ \forall i \neq j$. So,

$$\begin{aligned}
H(X \mid \widehat{X}=i) &= -(1-D)\log(1-D) - \sum_{i\neq j}\frac{D}{M-1}\log\left(\frac{D}{M-1}\right) \\
&= -(1-D)\log(1-D) - D\log\left(\frac{D}{M-1}\right) \\
&= h(D) + D\log(M-1)
\end{aligned}$$

Substituting,

$$\begin{aligned}
I(X;\widehat{X}) &= \log m - h(D) - D\log(M-1) \\
R(D) &= \log m - h(D) - D\log(M-1)
\end{aligned}$$

Digression:

What happens, if we use scalar quantizer for the above mentioned system (quantize $X \in \{1, 2, \ldots\ldots, M\}$)?

Suppose we use an uniform quantizer.

$$\left|\underbrace{1,\ 2,}_{\leftarrow\Delta\rightarrow}\right|\left|\underbrace{\ldots\ldots}_{\leftarrow\Delta\rightarrow}\right|\ldots\ldots\left|\underbrace{,\ M-1,\ M}_{\leftarrow\Delta\rightarrow}\right|$$

The reconstruction points will be $i\left(\frac{\triangle+1}{2}\right)$ for i=1,3,....M-1 odd no.s

Find average distortion D: This will be same for each points (uniform quantizer). We are using hamming distortion. $d = \begin{cases} 1 & \Delta \neq i \\ 0 & \Delta = i \end{cases}$. Thus,

$$\begin{aligned} D &= \frac{\Delta - 1}{\Delta}.1 + \frac{1}{\Delta}.0 \\ &= \frac{\Delta - 1}{\Delta} \end{aligned}$$

Rate-Distortion trade-off: we have $\frac{M}{\Delta}$ possible outputs. So, $\log \frac{M}{\Delta}$ number of bits. Then,

$$R(D) = \log \frac{M}{\Delta}$$

But, $D = \frac{\Delta - 1}{\Delta} = 1 - \frac{1}{\Delta}$ implies, $\frac{1}{\Delta} = 1 - D$
Thus the rate-distortion for this scheme is,

$$\begin{aligned} R &= \log M(1 - D) \\ &= \log M - \log \frac{1}{1 - D} \end{aligned}$$

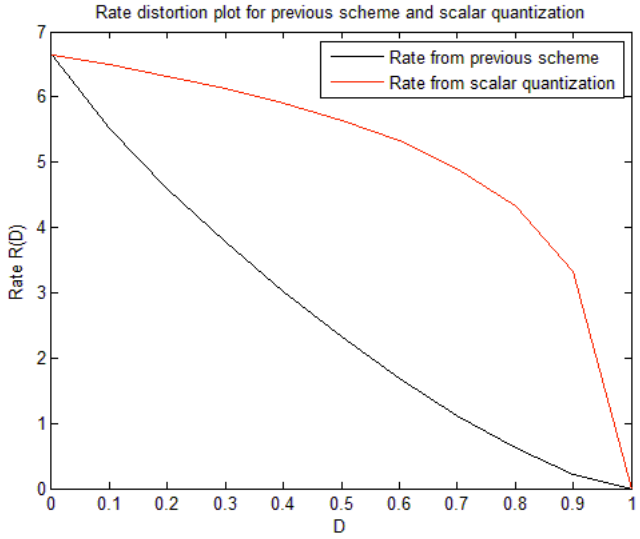

Figure 2: Rate distortion function and performance of and scalar quantization

From Figure 2 , it is evident that, the rate for scalar quantizer is always higher than the rate-distortion function. Thus, this scheme is suboptimal.

Kolmogorov Complexity

Introduction

So far, a given sequence of data (object) X was treated as a random variable with probability mass function $p(x)$. And the attributes (properties) defined for the sequence like entropy $H(X)$, average length $L(C)$, relative entropy divergence $D(p \parallel q)$, Rate-distortion $R(D)$ etc., depended on the probability

distribution of the sequence. Most of the coding techniques and quantization techniques that we saw, also depended on the probability distribution of the sequence. We can define a descriptive complexity of the event $X = x$ as $\log \frac{1}{p(x)}$. But, Andrey Kolmogorov (Soviet mathematician) defined an algorithmic (descriptive) complexity of an object X to be the length of the shortest binary computer program that describes the object. He also observed that, this definition of complexity is essentially computer independent. Here, the object X is treated as strings of data that enter a computer and Kolmogorov Complexity is analogous to Entropy of this sequence.

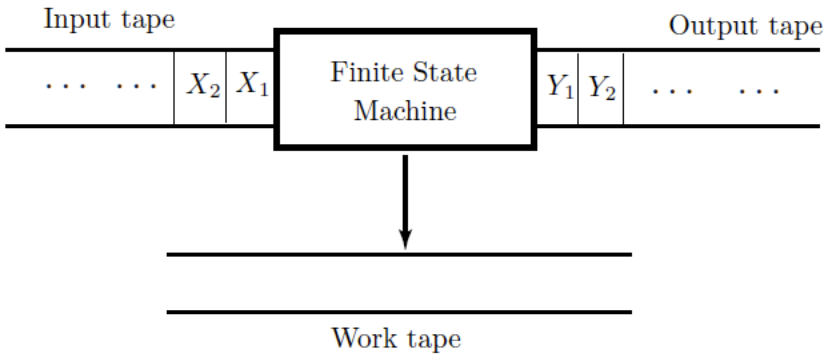

Figure 3: A Turing machine

An acceptable model for computers that is universal, in the sense that they can mimic the actions of other computers is 'Turing Machine' model. This model considers a computer as a finite-state machine operating on a finite symbol set. A computer program is fed left to right into this finite-state machine as a program tape (shown in Figure 2). The machine inspects the program tape, writes some symbols on a work tape and changes its state according to its transition table and outputs a sequence Y. In this model, we consider only a program tape containing a halt command (ie,. when to stop the program). No program leading to a halting computation can be the prefix of another program. This forms a prefix-free set of programs. Now the question is, given a string/sequence, can we compress it or not?

Answer: Kolmogorov Complexity and its properties.

Kolmogorov Complexity: Definitions and Properties

Definition:

The Kolmogorov complexity $K_{\mathcal{U}}(x)$ of a string x with respect to a universal computer $\mathcal{U}$ is defined as

$$K_{\mathcal{U}}(x) = \min_{p:\mathcal{U}(p)=x} l(p)$$

the minimum length over all programs that print x and halt. Thus, $K_{\mathcal{U}}(x)$ is the shortest description length of x over all descriptions interpreted by computer $\mathcal{U}$.

Conditional Kolmogorov complexity knowing $l(x)$ is defined as

$$K_{\mathcal{U}}(x \mid l(x)) = \min_{p:\mathcal{U}(p,l(x))=x} l(p)$$

This is the shortest description length if the computer $\mathcal{U}$ has the length of x made available to it.

Property 1:

If $\mathcal{U}$ is a universal computer, for any other computer $\mathcal{A}$ there exists a constant $\mathcal{C}$ such that,

$$K_{\mathcal{U}}(x) \leq K_{\mathcal{A}}(x) + \mathcal{C}$$

The constant $\mathcal{C}$ does not depend on x. All universal computers have same $K(x)$ so they differ by $\mathcal{C}$.

Property 2:

$$K(x|l(x)) \leq l(x) + c.$$

Conditional complexity is less than the length of the sequence. That is, the length of a program will be atmost length of our string x.

Example: "Print the following l length sequence : $x_1 \ldots\ldots x_l$"

Here l is given so the program knows when to stop.

Property 3:

$$K(x) \leq K(x|l(x)) + \log^*(l(x)) + c.$$

where $\log^*(n) = \log n + \log\log n + \log\log\log n + \cdots\cdots$

Here we do not know the length $l(x)$ of the sequence. The length of a sequence is represented as $\log l(x)$. But $\log l(x)$ is unknown. So, we need $\log\log l(x)$ and so on. Hence the term $\log^* l(x)$.

Property 4:

The number of binary sequence x with complexity $K(x) < k$ is $< 2^k$,

$$|\{x \in \{0,1\} : K(x) < k\}| < 2^k$$

The total length of any program of binary sequence is,

$$1 + 2 + 2^2 + 2^4 + \ldots\ldots\ldots + 2^{k-1} = 2^k - 1 < 2^k$$

Property 5:

The Kolmogorov complexity of a binary string X is bounded by

$$K(x_1 x_2 \cdots\cdots x_n \mid n) \leq nH\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) + \log^* n + c$$

Suppose our sequence X has 'k' ones. Can we compress a sequence of n bits with k ones? Given a table of X with k ones, our computer produces an index of length, $\log\binom{n}{k}$. But we do not know 'k'. So, to know 'k' we need $\log^* k$. So, the worst case length will be,

$$\log\binom{n}{k} + \log^* n + c$$

By Sterling's approximation,

$$\log\binom{n}{k} \leq nH\left(\frac{k}{n}\right) = nH\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right)$$

and we know, $K(X) \leq l(X)$,

$$K(x_1x_2\cdots\cdots x_n \mid n) \leq nH\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) + \log^* n + c$$

Property 6:

The halting programs form a prefix-free set, and their lengths satisfy the Kraft inequality,

$$\sum_{p:p \text{ halts}} 2^{-l(p)} \leq 1$$

Theorem:

Suppose $\{X_i\}$ is an iid sequence with $X \in \mathcal{X}$. Then,

$$\frac{1}{n}\sum_{x_1x_2\cdots\cdots x_n} K(x_1x_2\cdots\cdots x_n \mid n)\Pr(x_1x_2\cdots\cdots x_n) \longrightarrow H(X)$$

For large sequence, the Kolmogorov complexity approaches entropy.

Proof:

$$\begin{aligned}\sum_{x_1x_2\cdots\cdots x_n} K(x_1x_2\cdots\cdots x_n \mid n)\Pr(x_1x_2\cdots\cdots x_n) &\geq H(x_1x_2\cdots\cdots x_n)\\ &= nH(X)\end{aligned}$$

Here $K(x_1x_2\cdots\cdots x_n \mid n)$ is the smallest length for any program and because the programs are prefix free, this is the length of prefix-free codes. Then the LHS of above equation is nothing but the average length of the symbol. And we have $L(C) \geq H(X)$. Thus,

$$\frac{1}{n}\sum_{x_1x_2\cdots\cdots x_n} K(x_1x_2\cdots\cdots x_n \mid n)\Pr(x_1x_2\cdots\cdots x_n) \geq H(X)$$

next we have to prove this is less than $H(X)$.
From property 5, we have

$$\begin{aligned}\frac{1}{n}K(x_1x_2\cdots\cdots x_n \mid n) &\leq H\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) + \frac{1}{n}\log^* n + \frac{c}{n}\\ E\left\{\frac{1}{n}K(x_1x_2\cdots\cdots x_n \mid n)\right\} &\leq E\left[H\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right)\right] + \frac{1}{n}\log^* n + \frac{c}{n}\end{aligned}$$

$H(X)$ is a concave function, so using Jensen's inequality,

$$\begin{aligned}E\left\{\frac{1}{n}K(x_1x_2\cdots\cdots x_n \mid n)\right\} &\leq H\left[\frac{1}{n}\left(E\sum_{i=1}^{n} x_i\right)\right] + \frac{1}{n}\log^* n + \frac{c}{n}\\ &= H\left[\frac{1}{n}\left(\sum_{i=1}^{n} Ex_i\right)\right] + \frac{1}{n}\log^* n + \frac{c}{n}\\ &= H\left[E(X)\right] + \frac{1}{n}\log^* n + \frac{c}{n}\end{aligned}$$

but

$$\begin{aligned} E(X) &= 1.\Pr(x=1) + 0.\Pr(x=0) \\ &= \Pr(x=1) \end{aligned}$$

then

$$\begin{aligned} E\left\{\frac{1}{n}K(x_1x_2\cdots\cdots x_n \mid n)\right\} &\leq H\left[\Pr(x=1)\right] + \frac{1}{n}\log^* n + \frac{c}{n} \\ &= H\left[X\right] + \frac{1}{n}\log^* n + \frac{c}{n} \\ &\longrightarrow H(X) \end{aligned}$$

as $n \longrightarrow \infty$, Kolmogorov complexity approaches entropy

Incompressible Sequences

There are certain large numbers that are simple to describe like,

$$2^{2^{2^{2^2}}} \; or \; (100!)$$

But most of such large sequences do not have a simple description. That is, such sequences are incompressible. Given below is the condition for incompressible sequence.

A sequence $X = \{x_1x_2x_3\ldots x_n\}$is incompressible if and only if,

$$\lim_{n\to\infty} \frac{K(x_1x_2x_3\ldots x_n|n)}{n} = 1.$$

Thus, Kolmogorov complexity tells us given a sequence, how much we can compress. (Answering our question posted in the Introduction). That is, if $K(X)$ is of the order of length n then clearly, the sequence is incompressible.

Theorem:

For binary incompressible sequence $X = \{x_1, x_2, x_3, \ldots\ldots, x_n\}$,

$$\frac{1}{n}\sum_{i=1}^{n} x_i \longrightarrow \frac{1}{2}$$

i.e., approximately same # of 1's and 0's or the proportions of 0's and 1's in any incompressible string are almost equal.

Proof:
We have by definition,

$$K(x_1x_2x_3\ldots x_n|n) \geq n - c_n$$

where c_n is some number. Then by Property 5, we have

$$\begin{aligned} n - c_n &\leq K(x_1 x_2 \cdots\cdots x_n \mid n) \leq nH\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) + \log^* n + c \\ 1 - \frac{c_n}{n} &\leq H\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) + \frac{\log^* n}{n} + \frac{c}{n} \\ H\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) &\geq 1 - \frac{(c_n + c + \log^* n)}{n} \\ &= 1 - \varepsilon_n \end{aligned}$$

and $\varepsilon_n \longrightarrow 0$ as $n \longrightarrow \infty$,

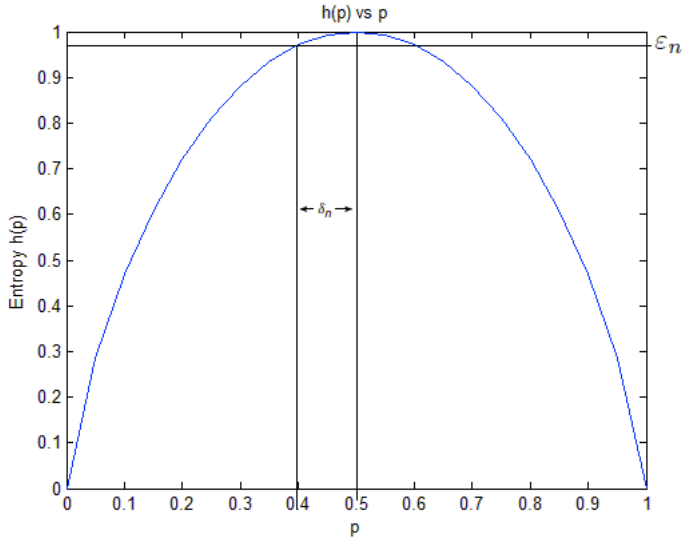

Figure 4: H(p) vs p

By inspecting the above graph, we see that,

$$\frac{1}{n}\sum_{i=1}^{n} x_i \in \left\{\frac{1}{2} - \delta_n, \frac{1}{2} + \delta_n\right\}$$

where δ_n is chosen such that,

$$H\left(\frac{1}{2} - \delta_n\right) = 1 - \varepsilon_n$$

this implies, $\delta_n \longrightarrow 0$ as $n \longrightarrow \infty$ and

$$\frac{1}{n}\sum_{i=1}^{n} x_i \longrightarrow \frac{1}{2}$$

References

[1] Chapter 14, "Elements of Information Theory" 2nd ed - T. Cover, J. Thomas (Wiley, 2006)

[2] http://en.wikipedia.org/wiki/Kolmogorov_complexity