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.

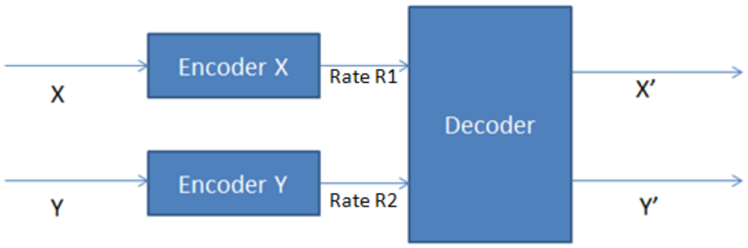


Figure 1: Block diagram for Slepian-Wolf coding: independent encoding of two correlated data streams X and Y

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.

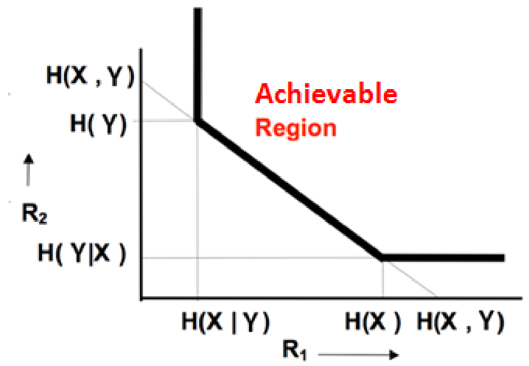
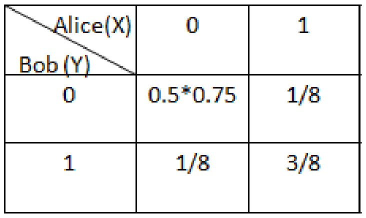


Figure 2: Achievable Rate Region

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.