Chapter 3

Source coding

3.1 Significance of Shannon Entropy

The Shannon entropy is the cornerstone in two complementary tasks of information processing: (i) file compression and (i) error correction. The Shannon **source coding theorem** – also called **noiseless coding theorem** – addresses the first issue: How much redundancy can be eliminated from a given source text without losing *any* information. The Shannon **channel coding theorem** – also called **noisy coding theorem** – addresses the second issue: How much redundancy must be added to a given source text in order to guarantee error free communication over a noisy channel.

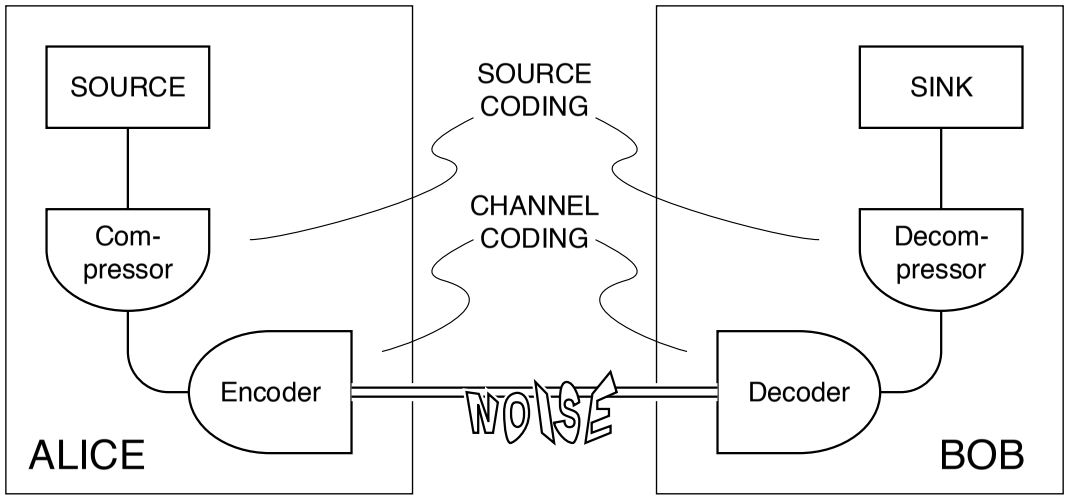

Figure 3.1: The generic model of a communication channel

3.2 Shannon source coding theorem

We attempt to encode strings

$$\mathbf{x} = x_1 x_2 \dots x_N \tag{3.1}$$

which consist of N symbols x_t, $t = 1, \dots, N$, each symbol $x_t \in \mathcal{A} = \{a_1, \dots, a_A\}$ being produced by a memoryless source $X = (\mathcal{A}, \mathbf{p})$ with probability $p(x_t)$. As there are A^N possible strings, the string $\mathbf{x} = x_1 \cdots x_N$ occuring with probability

$$P(\mathbf{x}) = p(x_1)p(x_2) \cdots p(x_N)\,, \tag{3.2}$$

if we take $2^{N\mathrm{ld}A}$ different codewords, each of length $N\mathrm{ld}A$ binary digits, we would have a perfect one-to-one block binary code.

The idea of the Shannon source coding theorem is not to insist on block encoding all strings, but only strings from some suitably chosen subset. The promise of the theorem is that if you are willing to sometimes fail in the encoding, because the string you are confronted with does not fall into this subset, you (1) will gain a lot, and (2) your failure tolerance ϵ can be as small as you chose – provided you accept to work with very long strings.

Your gain is in bits – instead of $H_0 = \mathrm{ld}A$ bits per symbol, which is required for the perfect block code, you will need only a tiny bit δ more than H bits per symbol, where $H = H(X)$ is the Shannon entropy of your source. The surprise of the theorem is that, asymptotically in the limit $N \to \infty$, this holds with true, with $\delta \to 0$, irrespective of your tolerance ϵ for encoding failures. In the end, if you are willing to deal with strings of length $N \to \infty$, you may even be absolutely intolerant to failures – i.e. your coding with only H bits per symbol, instead of the larger $\mathrm{ld}A$ bits per symbol, will succeed for sure.

Given your level of tolerance ϵ we will be dealing with a so called ϵ-sufficient subset S_ϵ, which is a subsets of strings such that $\mathrm{Prob}(\mathbf{x} \in S_\epsilon) \geq 1 - \epsilon$. This set contains $|S_\epsilon|$ strings, and we denote its raw-bit content $H_\epsilon(X^N) := \mathrm{ld}|S_\epsilon|$. A perfect binary block code for this set is obtained if we take $\frac{1}{N}H_\epsilon(X^N)$ bits per symbol.

For given ϵ there may be many different ϵ-sufficient subsets. We will be interested in the smallest ϵ-sufficient subset – if we have to live with failures, we want at least to spend as little resources as possible. A smallest ϵ-sufficient subset is easily constructed. Just order all the words from $\mathcal{A}^N$ left to right with decreasing probability. Start left, collect words until the sum of their probabilites exceeds $1 - \epsilon$. Done.

A seemingly different method to construct a ϵ-sufficient subset – not necessarily a smallest one – relies on the law of large numbers.[1] Consider $\frac{1}{N}\mathrm{ld}\frac{1}{P(\mathbf{x})}$, the Hartley

[1] This subsection is largly inspired by David J.C. MacKay's lecture notes "Information Theory,

information per symbol, of the word $\mathbf{x}$, which is drawn from X^N with probability $P(\mathbf{x})$. It is a random variable. It is in fact the normalized sum of N terms $\mathrm{ld}\frac{1}{p(x_t)}$, each of which is a i.i.d. random variable with mean $H = H(X)$ and square variance $\sigma^2 = \mathrm{var}[\mathrm{ld}(1/p(x)]$. We now define a set $T_{N\alpha}$, called the α-typical set,

$$T_{N\alpha} := \left\{ \mathbf{x} \in A^N | \left[\frac{1}{N} \mathrm{ld} \frac{1}{P(\mathbf{x})} - H \right]^2 < \alpha^2 \right\} \tag{3.3}$$

By the law of large numbers $\mathrm{Prob}(\mathbf{x} \in T_{N\alpha}) \geq 1 - \frac{\sigma^2}{\alpha^2 N}$, i.e. $T_{N\alpha}$ is a ϵ-sufficient subset with $\epsilon = \frac{\sigma^2}{\alpha^2 N}$. The definition of $T_{N\alpha}$ implies that, for all $\mathbf{x} \in T_{N\alpha}$, the probability of $\mathbf{x}$ satisfies

$$2^{-N(H+\alpha)} < P(\mathbf{x}) < 2^{-N(H-\alpha)} . \tag{3.4}$$

But how big is $T_{N\alpha}$? The smallest possible probability that a member of $T_{N\alpha}$ can have is $2^{-N(H+\alpha)}$, and the total probability of $T_{N\alpha}$ can not be any larger than one. Hence $|T_{N\alpha}| 2^{-N(H+\alpha)} < 1$, and thus the size of the typical set is bounded

$$|T_{N\alpha}| < 2^{N(H+\alpha)} \tag{3.5}$$

Recall that $T_{N\alpha}$ is ϵ-sufficient, with $\epsilon = \frac{\sigma^2}{\alpha^2 N}$, its size provides an upper bound for the size of the smallest ϵ-sufficient set,

$$H_\epsilon(X^N) < N \left(H + \frac{\sigma}{\sqrt{N\epsilon}} \right) . \tag{3.6}$$

Thus if you are ϵ tolerant to failures in encoding, you need less than $H + \frac{\sigma}{\sqrt{N\epsilon}}$ bits per symbol to encode any N-symbol string. For fixed ϵ, in the limit $N \to \infty$ this

Inference and Learning Algorithms", `http://wol.ra.phy.cam.ac.uk/mackay`.

upper bound coincides with the source entropy, and – moreover – this is true even if you become increasingly intolerant to errors, $\epsilon \to 0$.

We now show that if you attempt to use less than $H - \delta$ bits per symbol you will certainly fail in coding – there is no ϵ-sufficient subset which has less than $N(H-\delta)$ members. The proof is by contradiction. Assume that there is a ϵ-sufficient subset S' which has $N(H-\delta')$ members, $\delta' > \delta$. The probability to fall into this subset can be decomposed $P(S') = P(S' \cap T_{N\delta}) + P(S' \cap \overline{T_{N\delta}})$. The first term assumes its maximum if S' is all in $T_{N\delta}$, each word occuring with maximum probability $2^{N(H-\delta}$. The maximum value of the second term is $P(\mathbf{x} \neq T_{N\delta} < \sigma^2/(\delta^2 N)$. Hence

$$P(\mathbf{x} \in S') \leq 2^{N(H-\delta')} 2^{-N(H-\delta)} + \frac{\sigma^2}{N\delta^2} = 2^{-N(\delta'-\delta)} + \frac{\sigma^2}{N\delta^2}\,. \tag{3.7}$$

No matter how close positive $\delta' - \delta$ to zero, for N sufficiently large, it is impossible to have $P(S') > 1-\epsilon$, i.e. S' can not be ϵ-sufficient. Any subset S' with $|S'| < 2^{N(H-\delta)}$ has probability less than $1-\epsilon$, so by the definition of H_ϵ,

$$H_\epsilon(X^N) > N\left(H + \frac{\sigma}{\sqrt{N\epsilon}}\right)\,. \tag{3.8}$$

Combining () and () we obtain the following beauty

$$\forall_{\epsilon\in(0,1),\delta>0}\exists N_0 \forall_{N>N_0} \,:\, \left|\frac{1}{N}H_\epsilon(X^N) - H(X)\right| < \delta \tag{3.9}$$

What has just been said is nothing but the asymptotic equipartition principle: For an ensemble of N i.i.d. random variables, with N sufficiently large, the outcome $\mathbf{x} = x_1 \cdots x_N$ is almost certain to belong to a subset of $\mathcal{A}^N$ having only $\approx 2^{NH(X)}$ members, each occuring with probability close to $2^{-NH(X)}$.

With a little polishing, we thus have proved Shannon source coding

Theorem If a memoryless source X has entropy $H(X)$, then any uniquely decodeable code for this source into an alphabet of D symbols must have length at least $H(X)/\mathrm{ld}D$. Moreover, there exist such a uniquely decodeable code having average word length less than or equal to $1 + H(X)/\mathrm{ld}D$.

For the most important case of a binary symbol code, we may reformulate: For a source X with alphabet $\mathcal{A}$ and Shannon entropy $H(X)$, there exists a prefix binary symbol code C whose average word length satisfies

$$H(X) \leq L(C, X) < H(X) + 1\,. \tag{3.10}$$

For a complete proof, including the upper bound, see Dominic Welsh, *Codes and Cryptography*, Oxford University Press, N.Y., 1998. Informally what the theorem says is that if your source produces A different symbols, $A = 4$, say, in which case the raw bit content would be 2bit, but the source has only 0.62bit entropy, you may find a code f which maps the 4 source symbols into strings (codewords) formed from a $D = 2$ binary alphabet, such that the average length of the codewords is less than 1.62bit.

Note that the Shannon noiseless coding theorem only states the *existence* of efficient codes. It does not provide a *recipe* for producing such codes.

3.3 Huffman coding algorithm

A common recipe for producing efficient codes is the **Huffman coding algorithm**. The algorithm constructs recursively a succession of sources $X, X', \ldots, X^{(m-2)}$, where $X^{(k)}$ is obtained from $X^{(k-1)}$ by identifying the two least probable words of $X^{(k-1)}$ with a single word w in $X^{(k)}$. The probability that w is produced by $X^{(k)}$ is taken to be the sum of probabilities of its two constituent words in $X^{(k-1)}$.

 ©Martin Wilkens

Encoding proceeds backwards, by first encoding $X^{(m-2)}$, then $X^{(m-3)}$ and so on, until finally encoding X. The working of the Huffman algorithm is best described by an example:

X			X'			X''		
w	p	f	w'	p'	f'	w''	p''	f''
a	0.9	1	a'	0.9	1	a"	0.9	1
b	0.05	01	b'	0.05	01	b"	0.1	0
c	0.025	001	c'	0.05	00			
d	0.025	000						

The Huffman algorithm produces an optimal symbol code for a given ensemble. Yet it is not used in practice. The reasons are (i) the ensemble is fixed, and (ii) the extra bit.

When compressing text files, the character frequencies vary with context: a theoretical physics text has many more "H" than an experimental physics text – the reason being the theorist obsession with Hamiltonians. Huffman coding of a theoretical physics text using the standard relative frequencies of characters in english texts thus assumes the "wrong" probability distribution which according to Kullback-Leibler leads to quite some overhead in waste bits.

The innocuous-looking extra bit (relative to the ideal average length of $H(X)$) makes problems for small symbol alphabets, because in this case the overhead $L(C,X) - H(X)$ may dominate the length of the encoded file. A way out would be to compress blocks of symbols, for example the extended sources X^N. As the overhead per block is at most 1bit, the overhead per symbol would be $1/N$bits, which for sufficiently large blocks $N \gg 1$ would be negligeable. However the price to pay would be (i) to lose instantaneous decodeability, and (ii) to be forced to compute the probability of all strings (most of which will never occur) and build the associated Huffman tree.

Finally, for practical applications like the compression of text-files, the assumption of a memoryless source (of characters) does not account for the context-dependent correlations between characters, which dominate natural languages, however. The occurrence of a "q" before a "u" is much more likely than before a "x", say.

Summarizing, Huffman codes are of quite some theoretical importance, but do not play much of a role in applications. For practical applications so-called *stream codes* are widely used. These are codes where the symbol probabilities are modelled dynamically, i.e. while reading the source – see the supplement 3.4 for more.

3.4 Supplement: Codes – an overview

A code is a one-to-one map $f : \mathcal{S} \to \mathcal{C}$, where $\mathcal{S}$ is a set of words over and alphabet $\mathcal{A}$, and $\mathcal{C}$ is a set of words over an alphabet $\mathcal{D}$. For $\mathbf{w} \in \mathcal{S}$ the image $f(\mathbf{w})$ is called a *code word*.

Quite often a code is identified with the set of code words $\mathcal{C} = \{f(\mathbf{w})|\mathbf{w} \in \mathcal{S}\}$. If all code words of a code have equal length the code is called *block code*. Otherwise it is called *variable-length* code. Other types of codes are symbol codes and stream codes.

In symbol codes, each source symbol is assigned a code word, typically of variable length. The most prominent example is the Huffman code. The Huffman code is a compression code, that is it eliminates redundancy.

In block codes, source words of a given length, K say, are assigned the code words of a given length L, say. The most prominent example is the (binary) Hamming code. The Hamming code is an error correction code – it adds redundancy in order to cope with channel noise.

In stream codes both the pre-images (encoding) as well as the images (decoding) of the code words are words of variable length. They rely on dynamic modeling of the source. The most prominent example of compression codes are the arithmetic coding and the Lempel-Ziv coding.

3.4.1 Symbol codes

A symbol code is a map $f : \mathcal{A} \to \mathcal{C} \subset \mathcal{D}^*$ where $\mathcal{A}$ is a finite alphabet of A symbols, and $\mathcal{D}^*$ denotes the collection of finite strings over $\mathcal{D} = \{d_1, \ldots, d_D\}$. The code is completely specified by A code words $c_i = f(a_i)$. The integers $l_i := |f(a_i)|$ are called the *word lengths* of $\mathcal{C}$. The average length of the code is defined by

$$L(\mathcal{C}, X) := \sum_{i=1}^{A} p_i l_i \tag{3.11}$$

The extension of a symbol code f to strings $\mathbf{x} = x_1 x_2 \cdots x_n$, $\mathbf{x} \in \mathcal{A}^*$, is defined by the concatenation $f(\mathbf{x}) = f(x_1)f(x_2)\cdots f(x_n)$.

Uniquely decodeable is a symbol code $\mathcal{C}$ if any finite string $\in \mathcal{D}^*$ is the image of at most one string.

Instantaneous is a symbol code if there do not exist distinct a_i and a_j such that $f(a_i)$ is a prefix of $f(a_j)$. Here, for $x, y \in \mathcal{D}^*$, x is a prefix of y if there exists $z \in \mathcal{D}^*$ such that $xz = y$. Instantaneous codes are also called *prefix* codes.

Example The family of binary symbol codes $C : \mathcal{A} \to \{0, 1\}^*$. For example $\mathcal{C}_1 := \{0, 101\}$ is instantaneous, but $\mathcal{C}_2 := \{1, 101\}$ is not.

3.4.2 Block codes

A (L, K) **block code** is a list of $M = 2^K$ codewords, each of length L. The rate of a (L, K) block code is $R = K/L$. Note that M and L must be integer, but $K = \mathrm{ld}M$ could be non-integer.

Hamming distance of two codewords $\mathbf{x}$, $\mathbf{y}$ of a block code is defined $d(\mathbf{x}, \mathbf{y}) =$ number of places where $\mathbf{x}$ and $\mathbf{y}$ differ.

Minimum distance $d(\mathcal{C})$ of a block code $\mathcal{C}$ is defined

$$d(\mathcal{C}) = \min d(\mathbf{c}_i, \mathbf{c}_j) \quad \forall i \neq j\,. \tag{3.12}$$

If a block code has minimum distance d, the minimum-distance decoding scheme can correct up to $(d-1)/2$ noise induced symbol flips.

For the important case of a binary block code, encoding is a rule which assigns a sequence (word) of K source bits $\mathbf{s}$ ("signal") a sequence (code word) of L code bits $\mathbf{t}$ ("transmitted"). As we need redundancy, we require $L > K$.

Example The set $\{\texttt{000}, \texttt{111}\}$ is a fixed-length binary block code of length $L = 3$, called the repetition code $R_3 = (3, 1)$. The rate of this code is $R_{R_3} = 1/3$.

3.4.3 Errors and Decoding rules

Error is the event "$\mathbf{t}^{(i)}$ transmitted, the received $\mathbf{y}$ decoded as $\mathbf{t}^{(j)}$, where $j \neq i$".

The error probability – or average probability of decoding error – is defined

$$e(\mathcal{C}; Z, D) = \frac{1}{M} \sum_{i=1}^{M} \mathrm{Prob}(\texttt{ERROR} | \mathbf{t}^{(i)} \text{ transmitted})\,. \tag{3.13}$$

where Z is the channel, and D is a *decoding rule*.

The maximum error probability is defined

$$\hat{e}(\mathcal{C}; Z, D) = \max_i \text{Prob}(\texttt{ERROR}|\mathbf{t}^{(i)} \text{ transmitted}) \tag{3.14}$$

The minimum-error rule – also called *ideal observer rule* – decodes a received $\mathbf{y}$ into a $\mathbf{t}^{(i)}$ such that

$$\text{Prob}(\mathbf{t}^{(i)} \text{ sent}|\mathbf{y} \text{ received}) \geq \text{Prob}(\mathbf{t}^{(j)} \text{ sent}|\mathbf{y} \text{ received}) \quad \forall j\,. \tag{3.15}$$

The rule looks fine, but is difficult to implement: you would need to know the probabilities with which the $\mathbf{t}^{(i)}$ are used.

The maximum-likelihood rule decodes a received $\mathbf{y}$ into a $\mathbf{t}^{(i)}$ that maximizes

$$\text{Prob}(\mathbf{y} \text{ received}|\mathbf{t}^{(i)} \text{ sent})\,. \tag{3.16}$$

In the case that the code words are equally likely, the maximum-likelihood rule agrees with the minimum-error rule.

The minimum distance rule decodes a received $\mathbf{y}$ into a $\mathbf{t}$ which has minimum Hamming distance to $\mathbf{y}$. For the binary symmetric channel with bit-flip probability $p_\text{b} \leq 1/2$, the minimum distance decoding rule is equivalent toe the maximum-likelihood decoding rule.

3.4.4 Linear Codes

Equipped with modulo 2 arithmetic, the set of all words $\{\texttt{0},\texttt{1}\}^L$ may be viewed a L-dimensional linear vector space over the Galois field $GF(2)$ (arithmetic modulo 2), called V^L.

A linear (L, K) **block code** is a fixed-length block code in which all the codewords span a K-dimensional subspace of V^L. Encoding can be represented by a $L \times K$ binary matrix G^T such that if the signal to be encoded, in binary notation, is $\mathbf{s}$ (a K-dimensional vector), then the encoded signal is $\mathbf{t} = G^T\mathbf{s}$ modulo 2.

In a *linear code*, the extra $L - K$ bits are linear functions of the original K bits. These extra bits are called *parity check bits*.

s	t	s	t	s	t	s	t
0000	0000 000	0100	0100 110	1000	1000 101	1100	1100 011
0001	0001 011	0101	0101 101	1001	1001 110	1101	1101 000
0010	0010 111	0110	0110 001	1010	1010 010	1110	1110 100
0011	0011 100	0111	0111 010	1011	1011 001	1111	1111 111

(7,4) Hamming code

[To be completed]

 ©Martin Wilkens