Information Channels

Suppose that data must be sent through a *channel* before it can be used. This channel may be unreliable, but we can use a *code* designed counteract this.

Some questions we aim to answer:

- Can we quantify how much information a channel can transmit?
- If we have low tolerance for errors, will we be able to make full use of a channel, or must some of the channel's capacity be lost to ensure a low error probability?
- How can we correct (or at least detect) errors in practice?
- Can we do as well in practice as the theory says is possible?

Error Correction for Memory Blocks

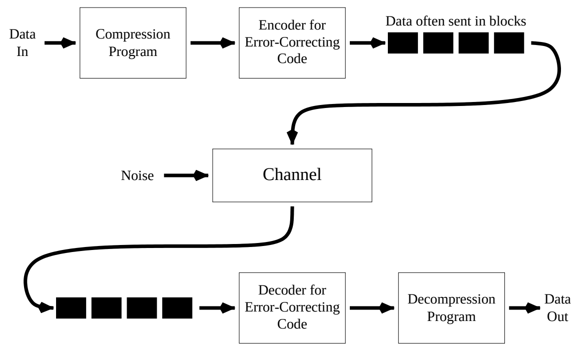
The "channel" may transmit information through time rather than space — ie, it is a memory device.

Many memory devices store data in blocks — eg, 64 bits for RAM, 512 bytes for disk.

Can we correct some errors by adding a few more bits? For instance, could we correct any single error if we use 71 bits to encode a 64 bit block of data stored in RAM?

Error Correction in a Communications System

In other applications, data arrives in a continuous stream. An overall system might look like this:


Data In
Compression Program
Encoder for Error-Correcting Code
Data often sent in blocks
Noise
Channel
Decoder for Error-Correcting Code
Decompression Program
Data Out

Error Detection

We might also be interested in detecting errors, even if we can't correct them:

- For RAM or disk memory, error detection tells us that we need to call the repair person.
- For some communication applications, we have the option of asking the sender to re-transmit.
- If we know that a bit is in error, we can try to minimize the damage — eg, if the bit is part of a pixel in an image, we can replace the pixel with the average of nearby pixels.

Formal Definition of a Channel

A channel is defined by

- An input alphabet of r symbols, called $a_1, \ldots, a_r$.
- An output alphabet of s symbols, called $b_1, \ldots, b_s$.
- A description of how the output depends on the input.

We will deal only with *memoryless channels*, in which each output symbol is influenced only by one input symbol. The behaviour of such a channel is defined by the *forward probabilities*:

$$P_{ij} \;=\; \Pr(b = b_j \,|\, a = a_i) \;=\; \Pr(b_j \,|\, a_i)$$

These are fixed by the physical nature of the channel. They can be changed only by redesigning it.

The Binary Symmetric Channel (BSC)

For the BSC, the input and output alphabets are both {0, 1}.

With probability P, the symbol received is the same as the symbol transmitted. With probability $1 - P = \overline{P}$, the wrong symbol is received.

We can view the input and output alphabets as Z_2, the field of integers modulo 2. The channel can then be seen as adding "noise" to the input:

$$b \;=\; a + n$$

where n is 0 with probability P and 1 with probability $1 - P$. The addition is modulo 2 (the same as exclusive-or).

The Binary Erasure Channel (BEC)

For the BEC, the input alphabet is {0, 1}, but the output alphabet is {0, 1, ?}. The "?" output represents an "erasure", in which the transmitted symbol is lost.

An erasure happens with probability $\overline{P}$; otherwise, the symbol is received correctly.

The forward probabilities for the BEC can be arranged in a *channel matrix*:

$$(P_{ij}) \;=\; \begin{pmatrix} P & 0 & \overline{P} \\ 0 & P & \overline{P} \end{pmatrix}$$

Channel Input Distribution

We can choose what input symbols we feed into the channel. We might send symbols from some source, the output of a data compression program applied to that source, or an error-correcting code for either of these.

For the moment, we'll assume that the symbols we put in are independent of each other, with some specified distribution:

$$p_i \;=\; \Pr(a = a_i)$$

We may aim to choose these *input probabilities* so that we make efficient use of the channel.

Deriving Some More Probabilities

The input and the forward probabilities together define the *joint probability* for any combination of channel input and output:

$$R_{ij} \;=\; \mathrm{Pr}\,(a = a_i \text{ and } b = b_j) \;=\; \mathrm{Pr}\,(a_i, b_j) \;=\; p_i P_{ij}$$

We can now find the *output probabilities*:

$$q_j \;=\; \mathrm{Pr}\,(b = b_j) \;=\; \mathrm{Pr}\,(b_j) \;=\; \sum_{i=1}^{r} R_{ij}$$

Finally, we get the *backward probabilities*:

$$Q_{ij} \;=\; \mathrm{Pr}\,(a = a_i \,|\, b = b_j) \;=\; \mathrm{Pr}\,(a_i \,|\, b_j) \;=\; R_{ij}/q_j$$

The backward probabilities give the situation from the receiver's point of view — given that I've received symbol b_j, how likely is it that the symbol sent was a_i?

Input, Output, and Joint Entropies

The amount of information being sent can be measured by the *input entropy*:

$$H(\mathcal{A}) \;=\; \sum_{i=1}^{r} p_i \log(1/p_i)$$

Similarly, the amount of "information" received (some of which may actually be noise) is measured by the *output entropy*:

$$H(\mathcal{B}) \;=\; \sum_{j=1}^{s} q_j \log(1/q_j)$$

We also have the *joint entropy*:

$$H(\mathcal{A}, \mathcal{B}) \;=\; \sum_{i=1}^{r}\sum_{j=1}^{s} R_{ij} \log(1/R_{ij})$$

This represents the amount of information available to an outside observer who knows what both the sender and the receiver know.

Mutual Information

We can now define the *mutual information* between the input and the output:

$$I(\mathcal{A}, \mathcal{B}) \;=\; H(\mathcal{A}) + H(\mathcal{B}) - H(\mathcal{A}, \mathcal{B})$$

The mutual information is meant to represent the amount of information that is being communicated from the sender to the receiver.

This makes intuitive sense: The difference of $H(\mathcal{A}) + H(\mathcal{B})$ and $H(\mathcal{A}, \mathcal{B})$ is the "overlap" in the knowledge of the sender and receiver — due to information having been transmitted.

But the real test of this definition is whether it leads to useful theorems and insights.