Goal

My goal is to prove rigorously the Shannon decoding theorem.

Let P_{XY} be the law of some channel from alphabet $\mathcal{X}$ to $\mathcal{Y}$ (we suppose that the two alphabet are the same size for convenience, but the proof is the same otherwise)

The capacity of the channel is defined as $C = \max_{P_x} I(X;Y)$

Theorem: if $R < C$, for any error rate δ, there exist n, and a encoder-decoder pair between $\{0,1\}^{\lfloor Rn \rfloor}$ and $\mathcal{Y}^n$ such that the probability of error is smaller than δ.

The first step to prove this theorem is to prove it is true when we allow the encoders to be generated randomly. We will justify afterwards why it proves the theorem for the non-random case.

Code

In the following code, `sample` is a function that takes an array `d` as argument, and returns `i` such that $P(\text{sample}(d) = i) = d[i]$. It draws from the distribution.

```
l = ... # size of canal alphabet
P_y_x = ... # list of lxl elements
P_x = ... # list of l elements summing to one

P_y = l*[None]
for sy in range(l):
    P_y[sy] = sum(P_y_x[sy][sx] for sx in range(l))

def entropy(distrib):
    return sum(p*log2(1/p) for p in distrib)

# compute H(Y|X) when X~d and y|x ~ dc
def conditional_entropy(d, dc):
    return sum(d[j] * entropy(dc[j]) for j in range(len(d)))

def X():
    return sample(P_x)

def Y(x):
    return sample(P_y_x[x])

def generate_encode_table(k, n):
    table = 2**k * (n * [None])
    for w in range(2**k):
        for j in range(n):
            table[w][j] = X()
    return table


def jointly_typical(n, x, y, epsilon):
    hx = entropy(P_x)
    hy = entropy(P_y)
    hxy = hx + conditional_entropy(P_y_x)

    log_px = sum(log2(P_x[x[j]]) for j in range(n))
    log_py = sum(log2(P_y[y[j]]) for j in range(n))
    log_pxy = sum(log2(P_x[x[j]] * P_y_x[x[j]][y[j]]) for j in range(n))

    return abs(hx - log_px/n) + abs(hy - log_py/n) + abs(hxy - log_pxy/n) < epsilon
```

```
def decode(k, n, table, y):
    for w in range(2**k):
        if (jointly_typical(n, table[w], y)):
            return w
    raise ValueError


def random_input(k):
    return [sample([1/2, 1/2]) for _ in range(n)]


def error(n, R, epsilon):
    k = int(n*R)
    table = generate_encode_table(k, n)
    input = random_input()
    x = table[x]
    y = [Y(s) for s in x]

    # will be useful in the proof
    hint = jointly_typical(n, x, y, epsilon)

    decoded = decode(k, n, table, y)
    return (input != decoded)
```

Make sure you understand what the code is doing before proceeding.

In the global variables,
- `p_y_x` is fixed because it is the property of the canal
- `l` is also a property of the canal
- `p_x` can be arbitrary. We chose the one that maximizes $I(X;Y)$

Analysis

We fix R and ε such that $R + \varepsilon < C$

Let P_n be the probability that `error(n, R, epsilon)` returns True.

My goal is to show that $\lim_{n\to\infty} P_n = 0$; i.e the probability of getting a decoding error tends to zero.

We decompose the error into two kinds: $P_n = P(\text{error} \cap \text{hint}) + P\left(\text{error} \cap \overline{\text{hint}}\right)$

The `hint` variable was not useful in the decoding scheme, it was introduced to better understand the proof. It represents whether the encoded sequence `x` and the transmitted sequence `y` were detected as jointly typical. Let's explore what jointly typical means.

Joint Typicality

Let's start by explaining what typicality means.

Let's say you know the law Q of some variable S.

The outcomes of S may be more or less likely, more or less *surprising*.

The surprisal of s is defined as $-\log_2(Q(s))$, and the entropy as the average surprise: $H(Q) = \mathbb{E}[-\log_2(Q(S))]$

Now, let's observe a sequence $\underline{S}$ of IID variables with the law Q

The surprise of $\underline{s}$ is calculated as $\sum_{i=0}^{n} -\log_2(Q(s_i))$

And in average, we expect the surprise to be n times the entropy. $\underline{s}$ is **typical** when its surprise is close to the average, i.e when

$| - \frac{1}{n} \sum_{j=1}^{n} \log P_{X(x_j)} - H(X)| < \varepsilon$

(of course, the notion of “typical” depends on a parameter ε)

And this concept is very useful. Let $A_{\varepsilon,n}$ be the set of all typical sequences of length n.

By the law of large numbers, the probability that $\underline{s}$ is typical goes to 1 as n goes to ∞: $P\left(A_{\varepsilon,n}\right) \approx 1$

In addition, since for every element of $A_{\varepsilon,n}$, it's probability is close to $2^{-nH(Q)}$, we can deduce the approximate number of elements in the set:

$\#A_{\varepsilon,n} \approx 2^{nH(Q)}$

That proves that most of the time, you need $\lceil nH(Q) \rceil$ bits to encode a $\underline{s}$

Now, let's talk about joint typicality.

We need a distribution P_X and a distribution $P_{Y|X}$. From those we can deduce P_Y

Let $\underline{x}, \underline{y}$ be 2 sequences of symbols. They are jointly typical when:

- $|\frac{-1}{n}\sum_{j=1}^{n} \log P_{X(x_j)} - H(X)| < \varepsilon_0$
- $|\frac{-1}{n}\sum_{j=1}^{n} \log P_{Y(y_j)} - H(Y)| < \varepsilon_1$
- $|\frac{-1}{n}\sum_{j=1}^{n} \log P_{XY}\left(x_j, y_j\right) - H(X,Y)| < \varepsilon_2$

With $\varepsilon_0 + \varepsilon_1 + \varepsilon_2 < \varepsilon$

Since we have 2 variables, we have to be vey careful about the dependency between them. We have different properties when:
- X_i and Y_i follow P_{XY} (**1**)
- X_i follows P_X and Y_i follows P_Y but they are independent. (**2**)

And that's exactly why joint typicality is useful for decoding. If we find x such that $\left(\underline{x}, \underline{y}\right)$ are jointly typical, that gives a hint that x may be the encoded symbol we want to find.

More precisely:

- if the sequence of couples (X_i, Y_i) is IID and follows P_{XY}, then the probability that $(\underline{X}, \underline{Y})$ is in the typical set goes to 0 as n goes to ∞.
- if in the contrary X_i is IID and follows P_X, Y_i is IID and follows P_y but X_i and Y_i are independent for every i, then the probability that $\underline{X}$ and $\underline{Y}$ are jointly typical is
  $\sum_{\underline{x}}\sum_{\underline{y}} \prod P_{X(x_i)} \prod P_{Y(y_i)} \mathbb{1}_{x \text{ and } y \text{ jointly typical}} \leq$
  $\sum_{\underline{x}}\sum_{\underline{y}} \frac{\prod P_{X(x_i)} \prod P_{Y(y_i)}}{\prod P_{XY}(x_i,y_i)} \prod P_{XY}(x_i,y_i) \mathbb{1}_{x \text{ and } y \text{ jointly typical}} \leq$
  $\sum_{\underline{x}}\sum_{\underline{y}} 2^{-nH(X)-nH(y)-nH(X,Y)+n(\varepsilon_1+\varepsilon_2+\varepsilon_3)} \prod P_{XY}(x_i,y_i) \leq 2^{-nI(X;Y)+n\varepsilon}$

The calculations are a bit cumbersome, but the conclusion is very important: knowing if $\underline{X}$ and ($\underline{Y}$ are jointly typical is a very good test to know if they are drawn according to P_{XY} or if they are independent.

End of the proof

$$P_n = \underbrace{P(\text{error} \cap \text{hint})}_{\text{collision}} + \underbrace{P\left(\text{error} \cap \overline{\text{hint}}\right)}_{\text{non typicality}}$$

Let's start with "non typicality":

$P\left(\text{error} \cap \overline{\text{hint}}\right) \leq P\left(\overline{\text{hint}}\right)$

If you look at the code, it is the probability that `x` and `y` are not jointly typical, with `y` drawn from `x` following `p_(Y|X)`. We saw that this probbility goes to zero (**1**).

Now, collision:

$P(\text{error} \cap \text{hint})$ is the probability that `decode` returns the wrong `w`. It happends when there exists a `w != input` such that `jointly_typical(n, table[w], y)` returns true.

We can decompose this probability depending on the value returned by `decode`:

$$\begin{aligned} P(\text{hint} \cap \text{error}) &= \sum_{w \neq \text{ input}} P(\text{hint} \cap \text{decode}(k, n, \text{table}, y) = w) \\ &\leq \sum_{w \neq \text{ input}} P(\text{jointly\_typical}(n, \text{table}[w], y)) \end{aligned}$$

And what is the relation between `table[w]` and `y` ? Since `w != input`, `x` and `table[w]` are independent, so `y` and `table[w]` are independent ! OMG !!!

That means we can use **(2)**: $P(\text{jointly\_typical}(n, \text{table}[w], y) \leq 2^{-nI(X;Y)+n\varepsilon}$

$P(\text{hint} \cap \text{error}) \leq \sum_{w \neq \text{ input}} 2^{-nI(X;Y)+n\varepsilon} \leq 2^k 2^{-nI(X;Y)+n\varepsilon}$

as a reminder:
- we chose P_X such that $I(X;Y) = C$
- $k = \lfloor nR \rfloor$ so $k \leq nR$

$P(\text{hint} \cap \text{error}) \leq 2^{n(R+\varepsilon-C)}$ so it decays exponentially.

—

We can now conclude: $P_n \to 0$

Non random case:

We know that for an average encoding table, the probability of error can be made inferior to δ

That means that there exists at least one encoding table with error probability inferior to δ, otherwise the average probability would be superior.

The proof is done □