

## 1 Chain rules for probabilities

For each of the 3 following chains of information, give the joint probability using the chain rule:

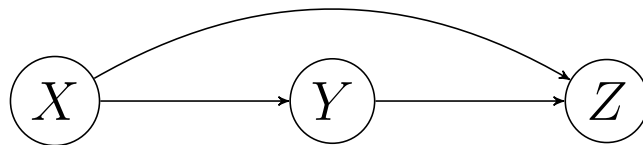
1.

$$p(X, Y, Z) = \dots?$$

$$p(X, Z) = \dots?$$

$$p(X, Y) = \dots?$$

$$p(Y, Z) = \dots?$$



2.

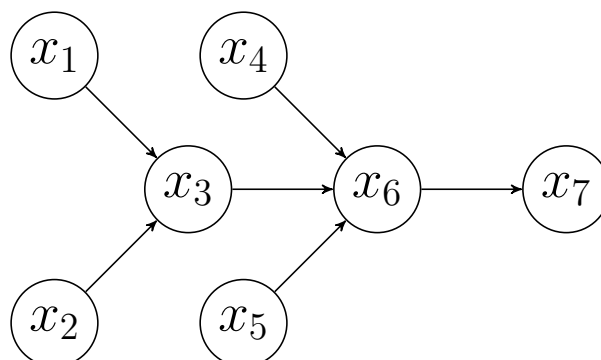
$$p(X_1, X_2, \dots, X_7) = \dots?$$

$$p(X_1, X_3, X_5, X_7) = \dots?$$

$$p(X_2, X_4, X_6, X_7) = \dots?$$

$$p(X_3, X_6, X_7) = \dots?$$

$$p(X_1, X_2, X_4, X_5) = \dots?$$



3.

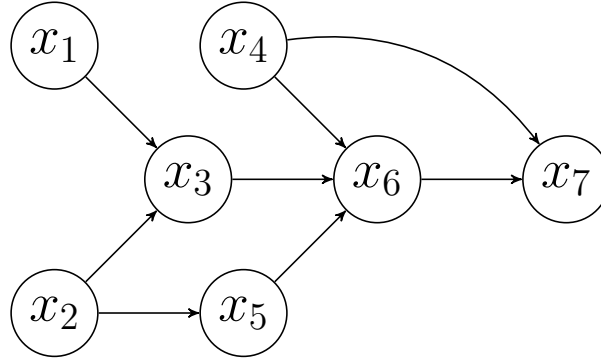
$$p(X_1, X_2, \dots, X_7) = \dots?$$

$$p(X_1, X_3, X_5, X_7) = \dots?$$

$$p(X_2, X_4, X_6, X_7) = \dots?$$

$$p(X_3, X_6, X_7) = \dots?$$

$$p(X_1, X_2, X_4, X_5) = \dots?$$



## 2 Communication through noisy channels

In this part, we are investigating the problem of data transmission through noisy channels and try to experiment with a very simple channel coding method to combat the errors.

Suppose you have an alphabet of 32 symbols and you want to communicate a message based on them, i.e. your (random) message consists of the concatenation of these symbols. Your communication channel is binary, so what you send to the channel as the representative of each symbol of the alphabet should be a binary sequence as well, i.e. they should consist only of '0's and '1's.

Because you want your communication to be as fast as possible, you want to have your binary sequences for each symbol as short as possible. As a natural way to choose these binary sequences, assign a 5-bit representative for each of the 32 symbols in the alphabet and keep them in a database (*in real-world scenarios more advanced coding is used, like [Huffman coding](#)*). Therefore, you will have necessarily all possible combinations of '0's and '1' in your database.

As the medium of communication, you have a **Binary Symmetric Channel (BSC)** with a probability of bit error equal to 0.1 (BSC,  $P_{be} = 0.1$ ). This means that every bit which is fed to this channel will be flipped in the output to its opposite due to noise, with a probability of 0.1. *You can view a simple implementation of BSC [here](#).*

1. Suppose you have a string of 1000 symbols that you want to send to your destination through this channel (you should use the same string for all experiments). Generate

the 5-bit binary representations of these symbols so that you will have a binary string of 5000 '0's or '1', and see the output of the BSC. Convert the output binary string to index numbers. Calculate the fraction of erroneous output symbols to the total number of communicated symbols (1000) by comparing the output string to the input string.

2. Redo the above experiment this time by generating binary representations of the symbols by repeating each bit 3 times so that you will have 15 bit representations for each symbol. As an example, suppose the binary representation of the 27th item in the database was '11010' in the previous part. Now by repeating each bit 3 times, this 27th item will be encoded as '111111000111000'. Send a string of 1000 symbols to the channel (a binary string of length 15000).

In the output of the channel, divide the received binary sequence into 5000, 3-bit long pieces and for each of these triples perform a majority vote between them. This means that if your received triple is '101', you should decode it as '1' because you have two '1's and one '0'.

Based on this encoding-decoding scheme, calculate the fraction of erroneous symbols to the total.

3. Redo the previous part this time by repeating 5, 11 and 23 times and derive the error fraction for each. Your code should have the ability to be easily generalized to different values of repetitions.
4. Calculate the elapsed time for transmission of the whole message. *Tip: you could use `timeit` magic command or see [this example](#) about using `timeit` module in python code.*
5. In Information Theory, to quantify how fast communication is, we define the rate of communication as:

$$R = \frac{\log_2 \mathbb{M}}{n}, (0 \leq R \leq 1)$$

where  $\mathbb{M}$  is the size of the alphabet and  $n$  is the length of the binary description of the words in the alphabet.

Calculate the rate of communication for each of the experiments above and compare it with the corresponding error fraction. Try to explain what happens when we increase the number of repetitions.

The above scenarios, follow essentially the general idea of **Channel Coding**, a very important branch of Information Theory with lots of achievements.

### 3 Moments

When  $X$  is a continuous real random variable of density  $f_X$ , one can define the moment-generating function  $M_X$  as:

$$M_X : t \in \mathbb{R} \mapsto \mathbb{E}_{f_X}[e^{Xt}] = \int_{x \in \mathbb{R}} f_X(x) e^{xt} dx$$

It is proven that the characteristic function defines the distribution. Moreover, when the moment-generating function  $M_X$  is integrable, we [can prove that](#):

$$f_X : x \in \mathbb{R} \mapsto \frac{1}{2\pi} \mathbb{E}_{\phi_X}[e^{-iXt}] = \frac{1}{2\pi} \int_{t \in \mathbb{R}} M_X(it) e^{-ixt} dt$$

using the [relation](#) between moment-generating and characteristic functions:  $\varphi_X(t) = M_{iX}(t) = M_X(it)$ .

The moment-generating function  $M_X$  also allows us to define the moments of  $X$  from its successive derivative values when they exist for  $t = 0$ . For any integer  $n \geq 1$ , the  $n$ -th moment of  $X$  is proven to be:

$$\mu_n = \mathbb{E}_{f_X}[X^n] = M_X^{(n)}(0) = \frac{\sum_{k=1}^{N_{\text{samples}}} x_k^n}{N_{\text{samples}}} \quad (1)$$

when  $N_{\text{samples}}$  of the distribution are given. As a consequence, the moment-generating function  $M_X$  can be written as the following Taylor series:

$$M_X(t) = \sum_{n=0}^{\infty} \frac{\mu_n}{n!} t^n$$

and can be approximated using the  $N$  first moments by

$$M_X(t) = \sum_{n=0}^N \frac{\mu_n}{n!} t^n + o(t^N) \quad (2)$$

## Questions

With `np.load('junction.npz')['data']`, load the grayscaled image and make it a flattened vector  $X$ . Generate a vector  $Y$  with the same dimension as  $X$ , whose values are randomly sampled from the Gaussian distribution of mean 0 and variance 1. For each of these two data, perform the following tasks:

1. Rescale the real data to be centered and between -5 and 5 and select only  $y \in Y: y \in [-5, 5]$  (square brackets denote **Indicator function**).

$$X \longleftarrow 5 \times \frac{X - \text{mean}(X)}{\max\{|X - \text{mean}(X)|\}} \quad Y \longleftarrow Y[Y \in [-5, 5]]$$

2. Display and save the **normalized** histogram of the data (to form a probability density).
3. Define and plot the moment-generating functions and approximated by:

$$M_X: t \in [-5; 5] \mapsto \sum_{j=1}^{N_j-1} f_X(x_j) e^{x_j t} (x_{j+1} - x_j)$$

where  $x_j, f_X(x_j)_{j=1}^{N_j}$  are the values given by the histogram, and:

$$M_Y: t \in [-5; 5] \mapsto \int_{x \in \mathbb{R}} \frac{e^{-\frac{x^2}{2}}}{\sqrt{2\pi}} e^{xt} dx$$

you can use `scipy.integrate.quad` function to compute the integral.

4. Find the 7 first moments ( $n = 0 \dots 6$ ) of these centered distributions given in Equation 1.
5. For each  $N = 0 \dots 6$  define and plot the approximation of the moment-generating function  $\phi_X(\cdot)$  as defined in Equation 2. You may need to plot them on a restrained domain such as  $[-1; 1]$  because of the errors of approximations far from 0.
6. Legend your figure accordingly.

Please describe your plot and comment your results (*simple pasting a figure is not enough*).

## Submission

Please archive your report and codes in “Prénom Nom.zip” (replace “Prénom” and “Nom” with your real name), and upload to “Upload TP6: Probabilities and communications.” on <https://moodle.unige.ch> before **Monday, November 27 2023, 21:59 PM**. Note, that the assessment is mainly based on your report, which should include your answers to all questions and the experimental results. *Importance is given on the mathematical explanations of your works and your codes should be commented*  
Please use this [overleaf template](#).