Topics in Information Theory / Home preparation

Exercises' questions

Rémi Peyre

29th November 2023

EXERCISE 1 — Exercising with logarithms and exponentials

Compute each of the following expressions when they are meaningful; or otherwise observe that they are meaningless. Computations shall be done with four significant digits. Quantities of information shall be expressed in whichever of the following units is the largest that yields a result of at least 1*:* nat*,* kio*,* Mio*,* Gio*,* Tio*; or in* dban *if they are less that* 1 nat*.*

1. $\log_6 242$.

2. $\mathrm{lb}\, 12345678$.

3. $\log 23456789$.

4. $\mathrm{colog}(1.661 \times 10^{-24})$.

5. $\lg(58\ \mathrm{ban})$.

6. 1 To.

7. 1 bit.

8. 3 ban + 8 o.

(9).★ $\log 7 \times 77$ bit.

10. $\exp(14)$.

11. $\exp(3.5\ \mathrm{ban})$.

12. $\mathrm{colog}(\exp(-262))$.

13. $\mathrm{coln}\exp(-\log 4567)$.

14. $\mathrm{colog}(10^{-100} \uparrow 100)$.

15.★ $\log \ln \lg(\mathrm{e} \uparrow 10^{100})$.

(16).★★ $\mathrm{lb} \lg \ln(2 \uparrow (\mathrm{e} \uparrow 10^{309}))$.

Topics in Information Theory

Rates of information for processes

Exercises' questions

Rémi Peyre

13th December 2023

EXERCISE 1 — Two models for English language

To tackle this exercise, first load the module `english.py`*. This module, among other things, defines a dictionary (in* Python*'s sense)* `BIGRAMS` *that tells, when you consider two consecutive letters in a text written in English (removing every space, punctuation, &c.), what the probability is that these two letters are resp.* `AA`*,* `AB`*, …,* `ZY`*,* `ZZ`*. (This data has been computed on a very large corpus made of all kinds of texts in English).*

1.❀ Using the values of `BIGRAMS`, explain how one can compute the conditional probability that, given that the letter you have just been reading is an `A`, the next letter will be a `B` (and likewise the probability that, conditionally to having read an `I`, the next netter will be a `N`, and so on, for all 26^2 possibilities). Create a dictionary of dictionaries, called `TRANSITIONS`, such that, for l, l' any two letters in $\{\texttt{A}, \dots, \texttt{Z}\}$, `TRANSITIONS`$[l][l']$ is the probability, after an l, that the next letter is l'.

Hint : Running `english.py` also creates the set `LETTERS`, which is $\{\text{`A'}, \dots, \text{`Z'}\}$, so that you may iterate on it.

Now, let us consider the Markov chain model *for English language, defined in the following way: it is a Markov chain on* $\{\texttt{A}, \dots, \texttt{Z}\}$ *whose transition probabilities are given by* `TRANSITIONS`*.*

2. Write a function `markovenglish` that takes three optional arguments called resp. `start` (default: None[*]), `length` (default: 1920) and `display` (default: True), and that does the following:

- It simulates a random realization of the Markov model for English on `length` time steps, starting from `start`;
- In the case `display` asks so, it prints the simulated sequence of letters onto the console, as a long non-quoted string of characters;
- In any case, it returns the simulated letters,[†] as a `length`-long list of 1-character strings.

[*]For the time being, the behaviour of the function is undefined when the `start` argument is None.

[†]When running `markovenglish` in the console, you may assign its output to '`_`' so that it does not get displayed.

Try that function and observe that, indeed, the results look somehow more like a real-life English text than if they were purely random sequences of letters.

3.❀ Given that all the 26^2 entries of `BIGRAMS` are nonzero (no need to check it! ☺), explain why it is obvious that our Markov chain will be irreducible, aperiodic, and positive-recurrent.

It follows from the above properties that our Markov chain has a unique equilibrium probability measure.

4.★ Explain how one can compute that equilibrium measure *without* solving a linear system. Store that equilibrium measure as a dictionary of frequences, called `EQUILIBRIUM`.

Hint : "Equilibrium" is something towards which you tend…

(5). Here there was actually a simpler way to compute `EQUILIBRIUM`: can you see which one? Check that it would yield the same result indeed.

6. Modify the function `markovenglish` so that, when the `start` argument is not specified (and then defaults to `None`), what you simulate corresponds to (some window of) the Markov processus under its *stationary* measure.

7.❀ Write a function `iidenglish` that works like `markovenglish`, except that it draws its letters in an i.i.d. way, according to the measure `EQUILIBRIUM`. (In this context, there is actually no need for a `start` argument).

8. Write a function `cologprob` whose first argument is a sequence of letters (input as a list of 1-character strings), whose second argument is either '`markov`' or '`iid`', and which computes the cologarithm (expressed in bans) of the probability of getting the first argument as a result if you simulate the process specified by the second argument (the number of letters drawn being implicit from the size of the first argument).

Hint : To compute decimal logarithms, use the function `log10` of the module `math` (whose functions are automatically loaded by `english.py`).

9.❀ Observe that, if the first argument is a very long sequence (take sequences of length $\sim 10^5$) generated randomly according to one of our processes, then, from run to run[‡], the result of `cologprob` is always almost the same, and that, if you change the length of the first argument, the result varies proportionnally to that length. From that, deduce that we have been computing the rate of some quantity relative to our processes; and give estimations of the four rates (with three significant digits) corresponding to the different possible combinations (two possibilities for the process generating the first argument, times two possibilities for the second argument).

10. The rates that we have been computing are rates of… what, actually?

[‡]Here it is implied that you keep everything constant: the process for generating the sequence in first argument, the length of that sequence, and the process w.r.t. whose law the colog-probability is computed: the only thing that changes from run to run is the randomness in drawing the first argument!

(In the forthcoming questions, I will call such rates by "rate of colog-probability according to the law [second argument of the function] for the process [first argument of the function]", so that it does not tell you the answer to the question above ☹).

11. In this context, the actual (mathematically exact) values of all our rates can be computed rather simply: so, compute them, and check that we find indeed the results found "experimentally"!

12. Among the four rates that you have been computing, you could know in advance that some equalities of inequalities would *necessarily* occur, from the simple fact that our two processes are different: which one(s)? (Your answer has to be complete: it must be the case that all the relations that you do not state may become wrong if we consider other processes than resp. the Markov and i.i.d. models for English!).

13.❀ Running `english.py` also generated a list of characters `AAIW`, which is an example of *actual* English text (viz, the text of *Alice's Adventures in Wonderland*, by Lewis Caroll). Use it to compute (the estimated value of) the colog-probability rate, according to resp. the Markov and the i.i.d. models, for the *real* English language.

14.★ So, we have been determining two extra rate values. Which inequalities, concerning these new rates, could you have told in advance that they would *necessarily* occur? Also, given that the Markov and the i.i.d. model are resp. an approximative model and a very approximative model for the English language, which other equalities or inequalities could you reasonably *expect*? (including inequalities between rates already discussed at Question 12).

EXERCISE 2 — Music on!

In Western classical music, the melody is built on a succession of chords[§]*, a chord being a set of notes that sound well together; and the strong notes in the main theme of the music have to belong to the chord that is currently active. Long story short, it is not unreasonable to model the succession of notes in a musical play as a* hidden Markov model*: the hidden process being the succession of chords, and the observed process being the succession of notes.*

In the module `music.py`*, you will find such a (very rough) model for music.* Notes *are represented by integers in* $[\![0, 12[\![$ *(0 being* C♮*, 1 being* C♯*, 2 being* D*, …, and 11 being* B*), the set of all notes (i.e.* $[\![0, 12[\![$*) being given by the variable* `ALLNOTES`*.* Chords*, on the other hand, are represented by* 2*-tuples, the first element of the tuple being a note (the* fundamental *note of the chord), and the second element being a character (either* `'M'` *or* `'m'`*) telling the* type *of the chord (resp.* major *or* minor*)—which yields 24 possible chords in total. The set of all chords is given by* `ALLCHORDS`*.*

The Markov chain kernel on chords is given by the dictionary of dictionaries `CHORD_TRANSITIONS`*. We will admit that this Markov kernel admits the uniform distribution (on the size-*24 *set of chords) as its equilibrium measure, and we will consider the stationary version of the Markov chain in the sequel. In our model,*

[§] In French: "accords".

for each chord, only one *note is played: this note is drawn randomly according to a probability distribution that depends on the active chord, the different probability distributions corresponding to the different chords being given by the dictionary of dictionaries* `NOTE_GIVEN_CHORD`.

1. Write a function `musichmm` that takes two optional argument called resp. `length` (default: 960) and `display` (default: `True`), and that does the following:

- It simulates a random realization of our HMM model for music notes (under its stationary measure) on `length` time steps;
- In the case `display` asks so, it prints the simulated sequence of notes onto the console, in human-readable form (see the hint);
- In any case, it returns the simulated notes, as a `length`-long list of integers of $[\![0, 12[\![$.

Hint : To make printing of notes more human-friendly, you can use the dictionary `NOTE2STRING` (created by `music.py`): for n a note-coding integer, `NOTE2STRING[`n`]` is the actual name of the coded note (which is always a 2-character string). By the way, there is also a similar dictionary `CHORD2STRING` to print the actual name of a chord (which is a 3-character string).

2.* Write a function `lawupdate` whose first argument is a dictionary repensenting a probability distribution on the set of all possible chords, the second argument is an integer representing a note, and which returns a dictionary representing a probability distribution on the set of chords, according to the following rule: is the first argument is P and that second is n, the output probability Q has to tell what is, from your point of view, the probability that the hidden layer of the HMM corresponding to note n is worth such or such chord, given that, prior to hearing note n, from your point of view, the probability that the *previous* chord (the one for the note *before* n) was such or such chord.

3.* Write a function `cologprob_musichmm`, whose argument is a sequence of notes (input as a list of integers), and which computes the cologarithm (in bans) of the probability of getting this result when simulating our HMM process (for the length corresponding to the argument given in input).

(4). Using the ideas of the previous questions, explain how one could simulate our HMM process without simulating the hidden layer as such. Implement such a simulation.

5.* From that, create a program that uses the Monte Carlo method to estimate the entropy rate of our HMM model for music, together with a (95 %) confidence interval. Apply this program to get an numerical estimate with a relative precision of 10^{-3}.

Hint : Beware that usual way to compute the confidence interval for the average of a probability distribution requires that the draws of that probability distribution are i.i.d.: but here; the notes of our process are not i.i.d...!

(6).** Prove that our HMM model is *not* Markov.

Hint : The reasoning of Questions 2 and 3 may be useful for this question...

Topics in Information Theory

Shannon entropy and message encoding

Exercises' questions

Rémi Peyre

10th January 2024

EXERCISE 1 — Implementation of the Shannon coding

The goal of this exercise is to get a better idea of the notion of Shannon entropy by implementing an efficient coding algorithm (namely, the Shannon codec) to real-world data. Here the data will be made by the text of Alice's Adventures in Wonderland*, the famous novel by Lewis Carroll.*[*]

Here we shall work on a simplified version of the text, where all letters were put in uppercase, and all punctuation marks and accents were removed, as well as all the page layout features and meta-information like chapter titles.[†] *Hence the novel appears as a string of* 107 087 *letters in the standard Latin alphabet* $\{\mathtt{A}, \mathtt{B}, \dots, \mathtt{Z}\}$ *(of cardinality* 26*).*[‡] *Actually, to make the text self-delimited, I also used the symbol* `*` *as a special character to mark the end of the text. This way, the novel becomes a self-delimited sequence of data on the* 27*-symbol set* $\{*, \mathtt{A}, \mathtt{B}, \dots, \mathtt{Z}\}$*, on which set we will consider the total ordering corresponding to the order of presentation that I wrote.*

The data corresponding to AAiW*'s text is given in the file* `Alice.txt`*. To use it in* Python*, first run the module* `Alice_auxiliary.py`*: this will (among other things) transcripting the contents of the text file into a list of single characters called* `ALICE_AS_LIST`*.*

First part

For the first two parts of the exercise, we assume that the characters from AAiW*'s text were drawn in an i.i.d. way. The probability distribution for the characters is taken as the empirical frequencies deduced from the data of the file* `English_monograms.tsv` *(which counted the occurrences of each letter in a very large corpus of English-language texts): we call it the* standard distribution *of the letters. Running the module* `Alice_auxiliary` *creates a dictionary* `LAW_OF_LETTERS` *giving the standard frequency for each character as a decimal number.*[§]

[*] Fun fact: Lewis Carroll was the pseudonym of Carles Dodgson, who was a mathematician in real life! ☺

[†] As regards *Alice's Adventures in Wonderland*, the page layout features are actually quite important…!

[‡] There are no digits in *AAiW*'s text; and if there had been some, I would have translated them in full letters.

[§] Actually the standard distribution given by `LAW_OF_LETTERS` is not *exactly* the empirical distribution of the corpus: in particular, we allowed for the possibility of the '`*`' character, rounded

For such a probability distribution on the set of characters, you can obtain the corresponding (interval) cumulated distribution function thanks to the function `cdf_from_law` *loaded by the* `Alice_auxiliary` *module. The ordering of the* 27 *characters used to compute that interval CDF is given by the list* `LIST_OF_CHARS`.[¶]

1.★ Write a function `digits_from_interval` that achieves the following: given two decimal numbers a and b as input, with $0 \leqslant a < b \leqslant 1$, it returns a string of decimal digits s such that all the numbers starting with $0.s\ldots$ lie in the interval $[a, b)$, and such that no shorter string of digits would have the same property.[‖] (For instance, for $a = 0.618\,033\,989$ and $b = 0.642$, the answers `6182` or `636` would *not* fit, because here you can find a string of length only two satisfying the property, namely `62`). (Note however that `63` would also be an acceptable return value).

2.❀ Create a string of digits that concatenates the results obtained from `digits_from_interval` applied successively to each of interval cdf values of the letters of *AAiW*'s text. Note down the length of that string.

3. Explain how someone that would be given the result of the previous question could build back the text of *AAiW* from it. (But do not implement that).

4. Would it be still possible to build back the text of *AAiW* from the string given at Question 2 if that string were followed by an infinite random sequence of digits, without knowing where the original string stops?

5. Compute the entropy of the standard distribution of letters, and show that it is worth approximately 1.272 ban.

6. According to the previous question, and given that the text of *AAiW* is 107 088 characters long (including the terminating '`*`'), what "should" be, approximately, the length of a relevant encoding of the text of *AAiW* into a string of decimal digits? (for the model of i.i.d. letters). Observe that we are far from that value here. What do you think is the *main* reason for this failure?… (Hint: look at the next questions ◡̈).

Second part

To try and improve on the problem detected above, we will group of characters by blocks of 10. *(Remark: that value is called* `BLOCKSIZE` *in* `auxiliary.py`*). The set of strings of length* 10 *is endowed with its lexicographic ordering (i.e., sorting is done according to the first character, then according to the second one in case of a tie, and so on). Note that it is a very big set (of cardinality* 27^{10}*), which could not fit inside your computer's memory!*

---

the frequencies to simple decimal values, and ensured that no character would be assigned zero probability.

[¶] Specifying the order as a list is important as *Python* provided no guarantee about the order in which the entries of a set or a dictionary are read within a `for` loop: actually even on the same computer is may change from one run to the other!

[‖] Using the vocabulary of Appendix A.2 in the textbook, this means that we are looking for the smallest possible value of k such that there exists an "aligned interval" of length 10^{-k} that is included in $[a, b)$; and that we have to find such an interval and to return the k first digits (following the decimal point) describing the numbers lying in this interval.

Our i.i.d. model predicts a probability for each of the possible 10*-character strings, which we may call the standard distribution on* 10*-letters blocks.*

7.★ Write a function `interval_from_string` that takes as input a block of 10 characters and returns the value of the interval cumulated distribution function for that block.

Hint : Actually it will be easier to understand first what would occur if you had blocks of 2 letters, then if you had blocks of 3 letters, &c.

8. Create a string of digits that concatenates the results obtained from `digits_from_interval` applied successively to each of interval cdf values of the 10-letters blocks of $AAiW$'s text (the last block being padded with extra '`*`' characters if needed).

9.★ Create a function `digits_to_string` that takes as input a string s of digits with length $\leqslant 30$ (macro `PREC_DEC`), which is supposed to be such that all the numbers starting with $0.s\ldots$ lie in the interval cdf value of a specific 10-letter string; and that returns (as a 2-uple), on the one hand, the value of that letter string, and on the other hand, the minimum numbers of digits that had to be taken into account to get that result.

10. Using the answer to the previous question, get back the text from $AAiW$ from the result of Question 8 (you may have that string followed by an arbitrary string of digits if needed).

Now let us tackle a few questions concerning the length of our encoded text.

11.❀ What is the number of digits per (text) character of the string obtained at Question 8? Compare it with the entropy of the standard distribution of letters.

12. What is the *empirical* distribution of the letters of $AAiW$, and what is the entropy of that distribution?

Hint : You should find about 1.252 ban.

13. For $n \gg 1$, consider an i.i.d. sequence of characters drawn according to the empirical distribution in $AAiW$; and assume that you try to encode that sequence *as if it had been drawn according to the standard distribution.* (Here self-delimitation is provided by the fact that you know the value of n: so, the character '`*`' has no special meaning). Show that, if you use the entropic coding as efficiently as possible, the encoded length should be of about $1.259\,n$. Could one expect that $1.259 < 1.272$? Could one expect that $1.259 > 1.252$? Could one expect that all these values would be fairly close to each other?…

(14). What would have been, approximately, the number of digits in our encoded text if we had had no information at all about the frequencies of letters in English? Comment on that.

Third part

Now for the last part of the exercise, we assume that the letters from AAiW*'s text are drawn according to a* Markov chain*, whose transition probabilities are drawn according to the dictionary of dictionaries* `LAW_OF_TRANSITIONS` *created b the module* `Alice_auxiliary`*. As regards the distribution of the first character of the text, we make as if it were preceded by a '*`*`*' character.*

15.* Write encoding and decoding algorithms for that case.

16.* Compute the stationary distribution for the law of transitions.

17. Deduce that the entropy rate for a string following our model shall be of 1.108 ban per character.

18. Re-do the two questions above for the *empirical* transitions of `AAiW` text (taking the fact that the first character is `A` as a transition from `*` to `A`). By the way, is the stationary distribution of the empirical model the same as the empirical distribution for the i.i.d. model? Could it be expected?

19. What would be the encoded length of a long text drawn according to the Markov model with the empirical probabilities of $AAiW$, if we had encoded it (with a sharp entropic encoding) according to the standard transition probabilities? Compare this value with the two values found in the questions above, and comment.

(20). Take a random sequence of 4096 i.u.d. digits and decode it according resp. to the i.i.d. and Markov models (terminate decoding when it is not possible to find the next block of characters). Which decoded string has the shortest length? Which decoded strings looks more like an actual English text? Were these properties expected?

Topics in Information Theory

Information theory, coding and statistics

Exercises' questions

Rémi Peyre

17th January 2024

EXERCISE 1 — A toy example for Shannon coding

Consider the following Markov chain, valued in $\mathbb{N}^*$*:*

- X_0 *is drawn according to the law* $\mathrm{Geom}(1/2)+1$ *(as regards the geometric distribution, see the hint);*
- *For all* t*, for all* $n>1$*,* $\mathrm{Law}(X_{t+1} \mid X_t = n) = \mathrm{Geom}(1/n)+1$*;*
- *The Markov chain is stopped as soon as it attains the value* 1 *(viz., denoting by* τ *the stopping time* $\min\{t\in\mathbb{N} \mid X_t = 1\}$*, we consider that* X_t *is actually not defined for* $t>\tau$*).*

Such a Markov chain describes a law on "words" over the alphabet $\mathbb{N}^*$*, that we may denote by* $\boldsymbol{P}$*.*[*]

1. Compute a possible encoding of the word "$2,5,4,1.$" over the alphabet {`A`, `B`, `C`, `D`, `E`} for the Shannon code associated to distribution $\boldsymbol{P}$ (using the standard orders on resp. $\mathbb{N}^*$ and {`A`, `B`, `C`, `D`, `E`}.

Hint : Here we "remind" that, for all $p\in[0,1)$, $\mathrm{Geom}(p)$ is the probability distribution on $\mathbb{N}$ such that $\mathbb{P}(\mathrm{Geom}(p)\geqslant n) = (1-p)^n$ for all $n\in\mathbb{N}$. Note that, as a consequence, the law $\mathrm{Geom}(p)+1$, which is deduced from $\mathrm{Geom}(p)$ by just adding one, satisfies the property that $\mathbb{P}(\mathrm{Geom}(p)+1>n) = (1-p)^n$ for all $n\in\mathbb{N}$: as, when dealing with integers, the events $\{X\geqslant n\}$ and $\{X+1>n\}$ are identical.

EXERCISE 2 — Fisher information rate for a hidden Markov model

For $\theta\in(0,1) =: \Theta$*, consider the (homogeneous) Markov chain* $(X_t)_{t\in\mathbb{N}}$ *on* $\{+1,-1\}$ *characterized by*

$$\begin{cases} \mathrm{Law}(X_0) = \mathrm{Unif}(\{+1,-1\}); \\ \mathrm{Law}(X_{t+1} \mid X_t = x) = (1-\theta)\delta_x + \theta\delta_{-x}: \end{cases}$$

in other words, at each step, the sign of X_t *switches with probability* θ *(resp. remains unchanged with probability* $(1-\theta)$*).*

[*] By the way, it is possible to show that in this case, the distribution $\boldsymbol{P}$ gives only mass to *finite* words: in other words, the Markov chain defined above does attain 1 in finite time almost-surely.

Also, independently of the chain $(X_t)_t$, consider i.i.d. random variables $(N_t)_{t\in\mathbb{N}}$ with $\mathrm{Law}(N_t) = 3/4 \cdot \delta_{+1} + 1/4 \cdot \delta_{-1}$ (i.e., N_t is a sign of $\{\pm 1\}$ which is negative with probability 1/4 (resp. positive with probability 3/4); and let

$$Y_t \coloneqq N_t X_t$$

for all $t \in \mathbb{N}$. We denote by $\mathbb{P}_\theta$ the corresponding probabilistic situation.

(1).❀ Explain why $(Y_t)_{t\in\mathbb{N}}$ is a hidden Markov model, and check that it is stationary.

Let $(y_{t\checkmark})_{t\in\mathbb{N}}$ be se sequence of $\{\pm 1\}^{\mathbb{N}}$, which we suppose to be known. For all $n \in \mathbb{N}$, let us denote respectively

$$\begin{aligned} p_t(\theta) &\coloneqq \mathbb{P}_\theta(X_{t-1} = +1 \mid \vec{Y}_{[\![0,t[\![} \equiv \vec{y}_{[\![0,t[\![\checkmark}); \\ q_t(\theta) &\coloneqq \mathbb{P}_\theta(X_t = +1 \mid \vec{Y}_{[\![0,t[\![} \equiv \vec{y}_{[\![0,t[\![\checkmark}); \\ r_t(\theta) &\coloneqq \mathbb{P}_\theta(Y_t = +1 \mid \vec{Y}_{[\![0,t[\![} \equiv \vec{y}_{[\![0,t[\![\checkmark}). \end{aligned}$$

2.★ Give the formulae to compute respectively

(i) $p_0(\theta)$;

(ii) $q_t(\theta)$ (as a function of $p_t(\theta)$);

(iii) $r_t(\theta)$ (as a function of $q_t(\theta)$);

(iv) $p_{t+1}(\theta)$ (depending on the value of $y_{n\checkmark}$, as a function of $p_t(\theta)$ and $q_t(\theta)$).

Hint : As regards $p_0(\theta)$, remember that stationary processes may actually be extended to the whole set of indices $\mathbb{Z}$…

Now, consider the statistical model whose hidden space parameter is Θ ('θ' being the symbol used to denote the values of the hidden parameter) and whose observation is $\vec{Y}_{[\![0,n[\![}$, the realization of that observation being $\vec{y}_{[\![0,n[\![}$. (Here, n is a (known) parameter of the model).

3. Express the colog-likelihood $\mathrm{coln}\,\mathscr{L}(\boldsymbol{\theta} = \theta \mid \vec{Y}_{[\![0,n[\![} \equiv \vec{y}_{[\![0,n[\![\checkmark})$ as a function of the $p_t(\theta)$'s, the $q_t(\theta)$'s and/or the $r_t(\theta)$'s.

4.★ Explain how you would compute efficiently, for a given $\theta_\checkmark \in \Theta$, the derivative at $\theta_\checkmark$ of the colog-likelihood of $\boldsymbol{\theta}$, that is,

$$\partial_\theta \,\mathrm{coln}\,\mathscr{L}(\boldsymbol{\theta} = \theta \mid \vec{Y}_{[\![0,n[\![} \equiv \vec{y}_{[\![0,n[\![\checkmark})_{\restriction \theta=\theta_\checkmark}.$$

Write all the formulae that would be needed to implement that computation numerically.

5. Same question for the second derivative

$$\partial_\theta^2 \,\mathrm{coln}\,\mathscr{L}(\boldsymbol{\theta} = \theta \mid \vec{Y}_{[\![0,n[\![} \equiv \vec{y}_{[\![0,n[\![\checkmark})_{\restriction \theta=\theta_\checkmark}.$$

In the sequel, as regards numerical applications, we shall use $\theta_\checkmark \coloneqq 0.2$.

6. Implement numerically all the previous computations (including the numerical values of $p_t(\theta = \theta_{\checkmark}), q_t(\theta = \theta_{\checkmark}), r_t(\theta = \theta_{\checkmark})$ for all $t \in [\![0, n]\!]$), for $n := 20$ and (denoting here merely '+' for $+1$, resp. '−' for -1) $\vec{y}_{[\![0,n[\![\checkmark} := (+,+,-,+,-,-,-,+,+,-,+,+,+,+,+,+,-,+,-,-)$.

Hint : You should find the following values for resp. the colog-likelihood (measured in nats) and its first and second derivatives: $13.806, -1.043, 25.662$.

For the last questions of this exercise, we take $n := 10^6$.

(7).❀ Implement and run a random simulation of the process $(Y_t)_t$ for $t \in [\![0, 10^6[\![$, and store its realization.[†]

For the next questions of this exercise, $\vec{y}_{[\![0,n[\![\checkmark}$ *refers to the realization of the above question's simulation.*

8. Explain how one can use the simulation of $\vec{y}_{[\![0,n[\![\checkmark}$ to get a good approximation of our model's Fisher information rate at $\theta = \theta_{\checkmark}$. (And, if you have done the implementations asked in the previous questions, compute that Fisher information rate).

Hint : You should find a rate of about 0.62.

(9).★★ Explain why we should <u>not</u> expect

$$n^{-1}\left|\partial_\theta \operatorname{coln}(\boldsymbol{\theta} = \theta_{\checkmark} \mid \vec{Y}_{[\![0,n[\![} \equiv \vec{y}_{[\![0,n[\![\checkmark})\right|^2$$

to be a good approximation of our model Fisher information rate. How could we nevertheless, using only the simulation $\vec{y}_{[\![0,n[\![\checkmark}$ and our implementations for the *first* derivative of the colog-likelihood, find a relevant approximation for the Fisher information rate?[‡]

Now, assume that you share your simulation $(\vec{y}_{[\![0,n[\![\checkmark})$ *with a colleague of yours, but that you do* <u>not</u> *tell the exact value* $\theta_{\checkmark}$ *that you used for* $\boldsymbol{\theta}$ *in this simulation. Your colleague wants to know the value of* $\boldsymbol{\theta}$*: to that end, she uses her skills in statistics to compute an estimation* $\hat{\theta}_{\checkmark}$ *for it. Here not that, given the large amount of data in your simulation, provided that your colleague's estimator was taken smartly enough (which we shall assume in the sequel),* $\hat{\theta}_{\checkmark}$ *will be fairly close to the actual* $\theta_{\checkmark}$*..*

10. Give the (approximative) lower bound on the root mean square error[§][¶] that your colleague shall commit on her estimation of $\boldsymbol{\theta}$, regardless of how smartly she chose her estimator.

[†] Here of course the $y_{t\checkmark}$'s for $t < 20$ have no reason to be the same as in the previous question! ☺

[‡] It should be noted that such an approximation, however, will generally be somehow less precise that the approximation using the second derivative (viz., the one of the previous question).

[§] Remember that the root mean square error (RMSE) refers to the square root of the average value of the squared error.

[¶] When speaking of "root mean square error", you have to imagine that you colleague *first* picks her estimation procedure, and <u>then</u> you draw $\vec{Y}_{[\![0,n[\![}$ randomly: so, here, we remember that $\vec{y}_{[\![0,n[\![\checkmark}$ was actually the realization of a *random variable*; and the notion of "mean" error refers to the randomness of that variable!

Hint : Here, if you did not compute it by yourself, you may use the value of the Fisher information rate that was given in the hint of Question 8.

Hint : You should that the RMSE is bounded below by about 1.6×10^{-3}.

11. Give an estimation procedure for which, according to the course, your colleague is guaranteed to attain (approximately) the optimal root mean square error.

12. Assuming that she uses a procedure that guarantees her to attain the optimal root mean square error (as found in the previous question), explain how your colleague can compute *by herself* (that is, still *without* knowing the exact value $\theta_{\checkmark}$) how much here root mean square error is.

13. Still assuming that she uses the procedure found at Question 11, if your colleague computes the 90 %-confidence margin of error for her estimator[!!], how much will that margin be worth (approximately)? Which specific result of the course did you use to get that answer?

Hint : We give the following values (rounded by above) for the quantile function Φ^{-1} of the standard normal distribution: $\Phi^{-1}(0.76) = 0.707; \Phi^{-1}(0.88) = 1.175; \Phi^{-1}(0.94) = 1.555$.

[!!] I.e., your colleagues wants to build a 88 % confidence interval centred at the estimator that she found; and she is interested in the half-width of that confidence interval.

Topics in Information Theory

Exercise: Description of continuous 2-dimensional data

Exercises' questions

Mines Nancy / Mathematical Engineering

7th February 2024

EXERCISE 1 — Description of continuous 2-dimensional data

During last session, we saw how to describe efficiently some continuous 1*-dimensional data up to some precision; and during today's lesson, I have explained the best way to describe multi-dimensional data according to some "purely geometric" criteria. Here we will again look for an efficient description of mutli-dimensional data; but the type of constraint on precision will be different compared to the lessons.*

The data that we are willing to described in model by a random variable, called $\boldsymbol{R}$*. The variable* $\boldsymbol{R}$ *is valued in* $\mathbb{R}^2$*; we call its coordinates resp.* $\boldsymbol{X}$ *and* $\boldsymbol{Y}$*. The law for* $\boldsymbol{R}$ *is denoted by* $\boldsymbol{P}$*; this law is supposed to have some density* f *w.r.t. the Lebesgue measure on* $\mathbb{R}^2$*. The function* f *is supposed to be smooth enough, and being* 0 *outside of a bounded set. We denote by* $\operatorname{supp} f$ *the* support *of* f*, i.e.* $\{r \in \mathbb{R}^2 \mid f(r) > 0\}$*; and we denote by* ℓ *a length expressing the typical order of magnitude of the fluctuations of* $\boldsymbol{R}$*.*[*]

Because our random variable is continuous, its realization cannot be described in full precision with only a finite amount of information length. Thus, our goal will be to describe it up to some small error, in a way as efficient as possible, in the sense that the approximated variable $\hat{\boldsymbol{R}}$ *should have minimal Shannon entropy. But, contrary to the course where our approximated variable had to be at some bounded distance to the true realization* almost-surely*, here we want to bound our error* in root mean square error*. In other words, if* $\hat{\boldsymbol{R}} =: (\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}})$ *is our approximation for* $\boldsymbol{R}$*, we want the mean square error*

$$RMSE(\hat{\boldsymbol{R}}) \coloneqq \mathbb{E}(|\hat{\boldsymbol{R}} - \boldsymbol{R}|^2)^{1/2}\text{[†]}$$

to be bounded above by some small length ε*. We will always consider that* $\varepsilon \ll \ell$*: so, we are interested in a description of* $\boldsymbol{R}$ *having* sharp *precision.*

In this problem, our general strategy to approximate $\boldsymbol{R}$ *will always be the same (see Figures 1 and 2):*

- *First we build a discrete subset* $\mathscr{D} \subseteq \mathbb{R}^2$*, where the points of* $\mathscr{D}$ *will typically be at distance of order of magnitude* ε *from each other.*

[*] To fix ideas, say that ℓ is the diameter of $\operatorname{supp} f$.

[†] For $r_0, r_1 \in \mathbb{R}^2$, $|r_1 - r_0|$ stands for the Euclidian distance between r_0 and r_1.

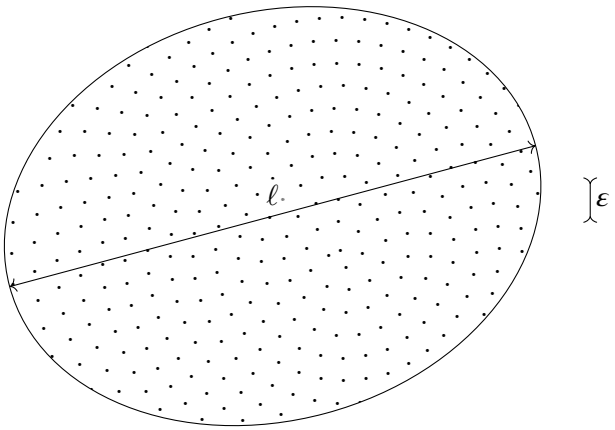

Figure 1: On this drawing, the ellipse would describe the support of f, i.e., the zone in which $\boldsymbol{R}$ may fall: ℓ being the typical size of this support, it morally gives the scale of the variations in the values of $\boldsymbol{R}$. On the other hand, ε, on the right of the drawing, is much smaller; so, it describes a scale for fine details of the variable $\boldsymbol{R}$. (In practice, the difference between ℓ and ε will be even larger—think of $\varepsilon / \ell \approx 10^{-7}$ or even 10^{-16}!—, but obviously this would have made the drawing impossible to understand…). The set $\mathscr{D}$ corresponds to the dots of the drawing, the typical distance between which being of the same order of magnitude as ε.

- *Then, for each eventuality ω, we approximate $\boldsymbol{R}(\omega)$ (i.e., the value of the random variable $\boldsymbol{R}$ for that eventuality) by the point of $\mathscr{D}$ which is closest to $\boldsymbol{R}(\omega)$ (breaking ties in an arbitrary way): denoting this point by $\hat{\boldsymbol{R}}(\omega)$, this defines indeed a random variable $\hat{\boldsymbol{R}}$ with values in $\mathscr{D}$ which will be a good approximation for $\boldsymbol{R}$.*

For the next questions, we will only consider what happens locally*: we focus on a little "mesoscopic" zone $\boldsymbol{U}$ whose size is $\ll \ell$ but $\gg \varepsilon$. Because that zone is $\ll \ell$, the density f is nearly constant in that zone: denote by $\boldsymbol{F}$ the "constant" value of $f_{\restriction \boldsymbol{U}}$. (More precisely, $\boldsymbol{F}$ is a value such that one has $f \approx \boldsymbol{F}$ on all $\boldsymbol{U}$).*

1. In the case when, inside $\boldsymbol{U}$, $\mathscr{D}$ follows the "square pattern" shown in Figure 3 (with δ corresponding to the distance shown on the drawing), show that, conditionnally to $\boldsymbol{R}$'s falling inside $\boldsymbol{U}$, one will have (up to negligible terms)

$$\mathbb{E}\big(|\hat{\boldsymbol{R}} - \boldsymbol{R}|^2 \;\big|\; \boldsymbol{R} \in \boldsymbol{U}\big) = \frac{\delta^2}{6}.$$

Note that, with the pattern of Figure 3, the "density" of points of $\mathscr{D}$ inside $\boldsymbol{U}$ is equal to δ^{-2}, in the sense that the number of points of $\mathscr{D} \cap \boldsymbol{U}$ is $\delta^{-2} \operatorname{vol}_2(V)$ (at first order).

2.❀ Show that, if the density of $\mathscr{D}$ inside $\boldsymbol{U}$ is ρ, and provided $\mathscr{D}$ makes a "regular" pattern inside $\boldsymbol{U}$ and its immediate neighbourhood, one will have

$$\forall r \in \mathscr{D} \cap \boldsymbol{U} \qquad \mathbb{P}(\hat{\boldsymbol{R}} = r) = \frac{\boldsymbol{F}}{\rho}$$

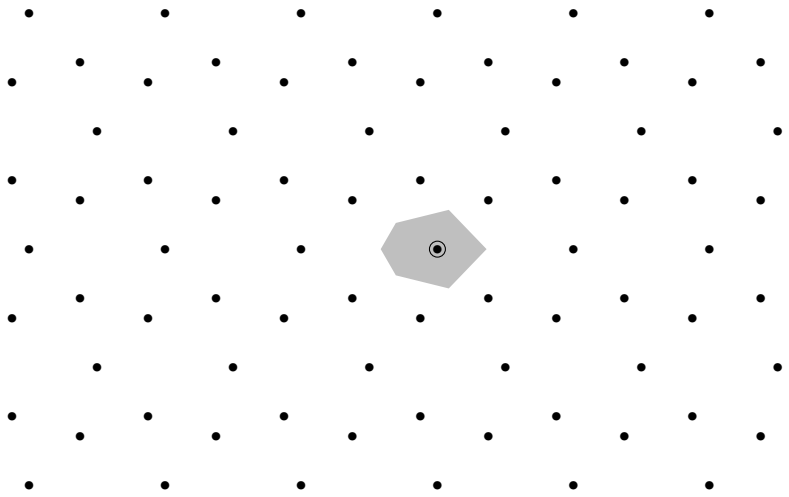

Figure 2: Here we supposed that we have strongly zoomed in a drawing like the one of Figure 1, each dot corresponding to a point in $\mathscr{D}$. The way of transforming $\boldsymbol{R}$ into $\hat{\boldsymbol{R}}$ consists in replacing $\boldsymbol{R}$ by the point of $\mathscr{D}$ which is closest to it: so, for instance, $\hat{\boldsymbol{R}}$ will be equal to the circled dot if and only if $\boldsymbol{R}$ falls inside the gray irregular hexagon.

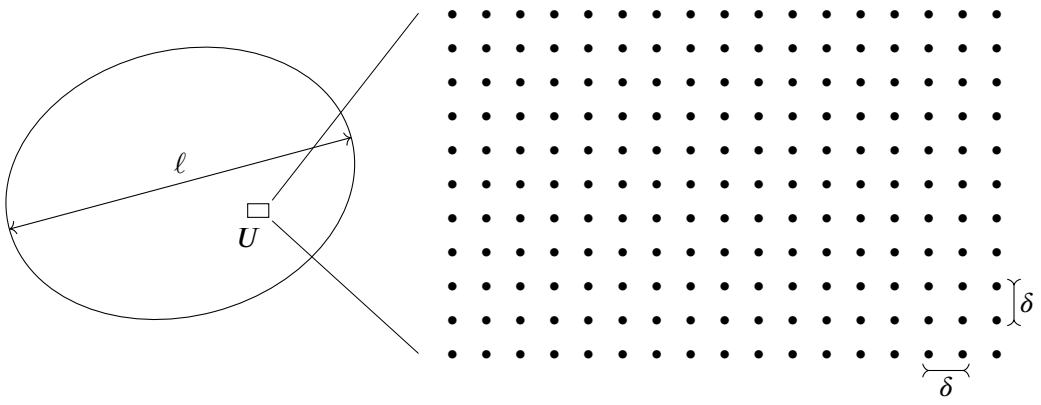

Figure 3: $\boldsymbol{U}$ is a mesoscopic zone, i.e., much smaller than the order of magnitude of the fluctuations of $\boldsymbol{R}$, whose typical length is ℓ (here, as in Figure 1, we assume that the ellipse is the support of the density of $\boldsymbol{R}$), but much larger than ε (and δ). The right part of the drawing represents a zoom on $\boldsymbol{U}$, on which we have represented the points of $\mathscr{D} \cap \boldsymbol{U}$. (Of course, in practice there would be much more many points of $\mathscr{D}$ in $\boldsymbol{U}$, and $\boldsymbol{U}$ would be much smaller w.r.t. ℓ: the ratios here are just to make the drawing understandible). When the points of $\mathscr{D}$ have the aspect on the right part of the drawing, we say that they follow the *square pattern*; the distance between two neighbouring points in the square pattern is denoted by δ—or $\delta(\boldsymbol{U})$ if the density of the pattern depends on $\boldsymbol{U}$. Because of what we told on the structure of $\mathscr{D}$, note that δ will always be of the same order of magnitude as ε.

Note that, given the answer to Question 1, for the pattern of Figure 3, the value of the density corresponding to having

$$\mathbb{E}\big(|R-\hat{R}|^2 \;\big|\; R\in U\big)=\varepsilon^2$$

is when you take a density of points of $\varepsilon^{-2}/6$.

3. From that, deduce that it is possible to describe the data R up to precision ε (in mean root square error) with $\hat{R}$ having an entropy of (up to some negligible terms)

$$-\log 6-\int_{r\in\mathbb{R}^2} f(r)\log(\varepsilon^2 f(r))\,\mathrm{vol}_2(\mathrm{d}r)$$

4. In the above formula, I chose not to develop "$\log(\varepsilon^2 f(r))$" into "$2\log\varepsilon+\log f(r)$", despite it would have yielded further simplifications… Could you explain why, in the current context, it would indeed not have been very good practice to make that expansion?

5. Show that, if we wanted to make ε twice smaller, we would have needed an extra average amount of information of 2 bit to describe R using that strategy. Why was this behaviour expected?

In the approach that we have been following, we used a constant density ρ *to get* $\mathbb{E}\big(|\hat{R}-R|^2 \;\big|\; R\in U\big)=\varepsilon^2$ *in all mesoscopic zones* U. *A savvy sudent suggests that we could make* ρ *vary so that* $\mathscr{D}$ *would be denser in zones where* f *is high, and sparser in zones where* f *is low. More precisely, she suggests, one would adjust* ρ *so that one would have (approximately) a constant value for* $\mathbb{P}(\hat{R}=r)$ *on the* whole $\mathscr{D}$ *(not just on local zones of* $\mathscr{D}$*). This students says that this idea seems logical because of the asymptotic equipartition property.*

6. Explain, in informal terms, what the asymptotic equipartition property means in general, and why it seems indeed to make the student's idea relevant.

7. Show that, with the idea of the savvy student, if we have the right to make the Shannon entropy $\mathscr{H}(\hat{R})$ of the approximated variable equal to h, the root mean square error that we will attain with this strategy is

$$RMSE(\hat{R})=\left(\frac{\mathrm{vol}_2(\mathrm{supp}\,f)}{6\exp(h)}\right)^{1/2}.$$

8. From that, deduce that the method proposed by the savvy student will actually always *less* efficient than our original idea (or just as efficient, but never more). (Too bad…! ∴)

Now we are using a different idea: instead of using the square pattern of Figure 3, we will use the triangular *pattern of Figure 4. In this case, one can show that the density is* $\frac{2}{\sqrt{3}}\zeta^{-2}$*, while the local mean squared error is*

$$\mathbb{E}\big(|\hat{R}-R|^2 \;\big|\; R\in U\big)=\tfrac{5}{36}\zeta^2$$

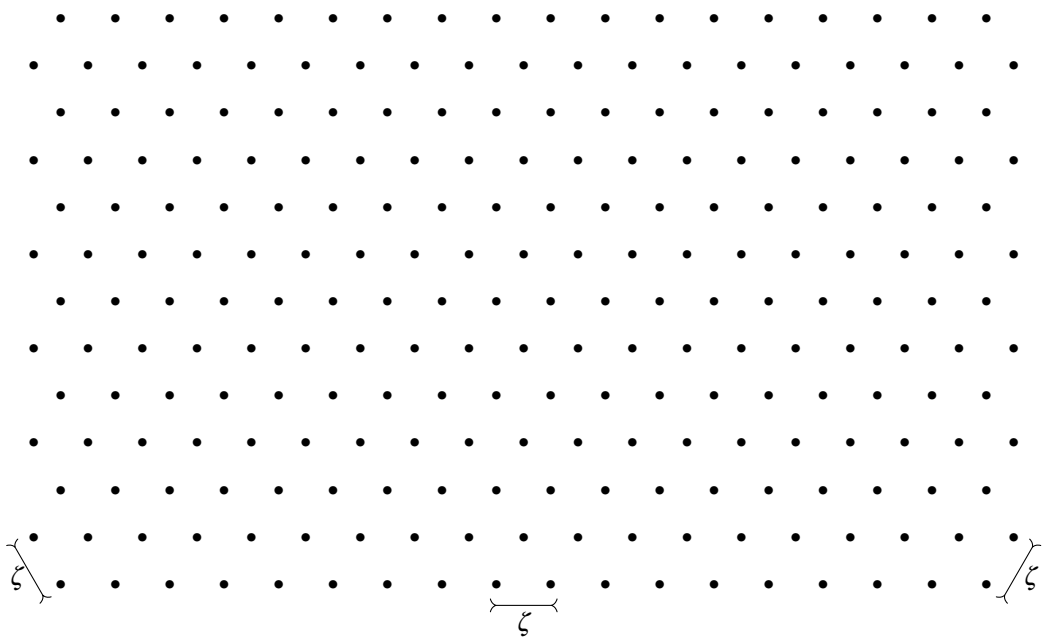

Figure 4: The triangular pattern, and what we call the length ζ for it. Here it is assumed that we have zoomed inside a mesoscopic zone $\boldsymbol{U}$.

9. With such a pattern, if one mimicks the strategy of Question 3, give the formula for the entropy to describe it up to precision ε (see the hint below). Show that this new entropy is actually *lower* than the one of Question 3. By how much have we improved entropy by using the triangular pattern? Compute the numerical application for the gain in entropy and comment on that.

Hint : You are not asked to re-do the computation of the entropy: just point out what changes between Question 3 and the current question, and explain why these changes shall translate into the formula that you will write.

Actually the strategy of Question 3 was tantamount to describing $\boldsymbol{X}$ *and* $\boldsymbol{Y}$ successively[‡], *since* $\mathscr{D}$ *appeared as a product set; while the strategy of Question 9 really relied on a* collective *description of* $\boldsymbol{X}$ *and* $\boldsymbol{Y}$*. There are other situations where it is better to describe data collectively rather than individually. In particular, this is the case when* encoding *a signal: the last question of this exercise will ask you to explain why.*

Let us introduce some context. For $q \in [\![1, +\infty[\![$*, if* X *is a finite set of cardinality* q*, it is actually possible to compute the* exactly optimal *way some to encode self-delimitedly, over the binary alphabet, some data drawn according to* $\mathrm{Unif}(X)$*: for instance, for* $q = 3$*, one could show that an optimal solution would consist in encoding one of the possible values by* `0`*, and the other two by resp.* `10` *and* `11`*: in which case the expected signal length would be* $\left(\frac{1}{3} \times 1 + \frac{1}{3} \times 2 + \frac{1}{3} \times 2\right)$ bit $= \frac{5}{3}$ bit.

[‡]Beware: I wrote "successively", not *separately*! Here the way to describe Y may make use of the knowledge of the realization of X (via the knowledge of the conditional law $\mathrm{Law}(Y \mid X = x_\checkmark)$), but fundamentally the strategy first describes X, and then Y.

In general, let us denote by L_q the (exactly) optimal expected signal length that you can attain when encoding the uniform distribution on a set of cardinality q (over the binary alphabet).

Now assume that, for some $n \in \mathbb{N}$, we are willing to describe a sequence of n variables that are i.i.d. uniform on $[\![0, q[\![$. A simple solution would consist in concatenating the n successive encodings of all our individual variables (which is legit, since we are considering self-delimited encodings), in which case the expected signal length would be nL_q). However, if we want to go "full power", we can use the fact that the sequence that we are willing to encode is actually a uniform variable over the set of all length-n sequences, which set has cardinality q^n: so, the exactly optimal achievable signal length would be L_{q^n}.

10.* Prove that, if q is not a power of 2, for all n large enough, the second strategy will be strictly better than the first one:

$$L_{q^n} < nL_q \quad \text{for } n \to \infty.$$

Hint : A first step consists in observing that, for q not a power of 2, L_q will never be exactly equal to $\log q$: why?…