

# Notes and Comments on S. Mallat's Lectures at Collège de France (2024)

Learning and generation using random sampling

J.E Campagne\*

Janv. 2024; rév. 13 avril 2024

---

\*If you have any comments or suggestions, please send them to `jeaneric DOT campagne AT gmail DOT com`

## Table des matières

<b>1 Foreword</b>	<b>6</b>
<b>2 Lecture of January 17th</b>	<b>6</b>
2.1 Introduction . . . . .	6
2.2 Digressions around two emblematic examples with GPT-4 . . . . .	9
2.3 What is the mathematical framework? . . . . .	15
2.4 Link between supervised and unsupervised training . . . . .	17
2.5 What are the issues? . . . . .	19
2.6 Panorama of the Mathematical Domain . . . . .	20
2.6.1 Modeling . . . . .	20
2.6.2 Sampling . . . . .	23
2.7 Some illustrations . . . . .	25
2.8 Course Outline . . . . .	29
<b>3 Lecture of January 24th</b>	<b>29</b>
3.1 Random Models: Why? . . . . .	29
3.2 The Monte Carlo Method . . . . .	31
3.2.1 Volume Calculation . . . . .	34
3.2.2 Why Not Perform a Deterministic Calculation? . . . . .	36
3.2.3 Quasi-Monte Carlo Method . . . . .	38
3.2.4 Limits of the Monte Carlo Method . . . . .	39
3.3 Probability Modeling, Learning/Inference . . . . .	40
3.3.1 Small Example: Spam Detection . . . . .	40

3.3.2	Learning: Finding the Best Approximation . . . . .	41
3.3.3	Inference . . . . .	43
3.3.4	Which type of modeling to choose? Examples . . . . .	43
<b>4</b>	<b>Lecture of January 31st</b>	<b>49</b>
4.1	Causal and Non-causal Schemes . . . . .	50
4.2	Markov Field . . . . .	51
4.2.1	Definition and Properties . . . . .	51
4.2.2	Two Generic Examples . . . . .	54
4.3	Independence Properties . . . . .	58
<b>5</b>	<b>Lecture of Feb. 7th</b>	<b>65</b>
5.1	Introduction . . . . .	65
5.2	Parameter Learning . . . . .	66
5.2.1	Fisher and Shannon Frameworks . . . . .	66
5.2.2	Optimization to find $\theta^*$ . . . . .	68
5.3	The linear case . . . . .	71
5.4	Score Matching . . . . .	73
5.5	Study of an example: $\phi^4$ potential . . . . .	75
<b>6</b>	<b>Lecture of February 14th</b>	<b>77</b>
6.1	Total variation distance . . . . .	78
6.2	The Hessian of $D_{KL}$ in the Linear Case . . . . .	81
6.3	Fisher Divergence (Score Matching) . . . . .	82
6.4	Rewriting of $I(p, p_\theta)$ . . . . .	83

6.5	Example: Linear Energy Case . . . . .	84
6.6	Convergence of Score Matching . . . . .	85
6.7	An Illustrative Example of the Score Matching Problem . . . . .	87
6.8	Conditions for Using Score Matching . . . . .	89
<b>7</b>	<b>Session of Feb. 28th</b>	<b>95</b>
7.1	Introduction . . . . .	95
7.2	The One-Dimensional Case . . . . .	96
7.3	Ergodicity of a Deterministic Transformation . . . . .	97
7.4	Uniform Measure on $[0, 1]$ . . . . .	100
7.5	Rejection Techniques . . . . .	104
7.5.1	Simple Version . . . . .	104
7.5.2	Version based on $q(x)$ . . . . .	105
7.5.3	Acceptance Probability . . . . .	106
7.6	Importance Sampling . . . . .	107
7.7	Beyond "Rejection": Progression of Ideas . . . . .	109
7.8	Markov Chain . . . . .	111
<b>8</b>	<b>Session of March 6</b>	<b>113</b>
8.1	Markov Chains: Example . . . . .	114
8.2	Invariant Law . . . . .	116
8.3	Reversibility Conditions . . . . .	117
8.4	Convergence towards the invariant distribution . . . . .	119
8.5	Spectral analysis viewpoint of the invariant measure . . . . .	122
8.6	Metropolis-Hastings Algorithm . . . . .	123

<b>9 Session of March 13</b>	<b>128</b>
9.1 Ornstein-Uhlenbeck Equation . . . . .	130
9.2 Denoising Problem . . . . .	132
9.3 Denoising Network . . . . .	134
9.4 Transition from Memorization to Generalization . . . . .	135
9.5 Opening towards some research directions . . . . .	139
9.5.1 Return to denoising: wavelet bases . . . . .	139
9.5.2 Denoising networks: What do they do? . . . . .	142
9.5.3 Results of numerical experiments . . . . .	143
9.5.4 Reflections on the generative model by score diffusion . . . . .	145
9.5.5 Case study: turbulence . . . . .	148
9.6 Conclusion of this year . . . . .	152
<b>10 NDJE. Quelques ajouts personnels</b>	<b>153</b>
10.1 NDJE. Hybrid/Hamiltonian Monte Carlo . . . . .	153
10.2 Normalizing Flows . . . . .	159

## 1. Foreword

**Disclaimer:** *In the following, you will find my notes taken in a free style as they come, and reshaped with some comments ("ndje" or dedicated sections). It is clear that errors may have slipped in, and I apologize in advance for them. You can use the email address provided on the cover page to send them to me. I wish you a good reading.*

Please note that on the website of the Collège de France, you will find all the videos of the courses, seminars, as well as the course notes not only for this year but also for previous years<sup>1</sup>.

I would like to thank the entire team of the Collège de France who records and edits the videos without which the editing of these notes would be less comfortable.

Also note that S. Mallat<sup>2</sup> provides free access to chapters of his book "*A Wavelet Tour of Signal Processing*", 3rd edition, as well as other materials on his ENS website.

This year, 2024, marks the seventh of the Data Science chair cycle by S. Mallat, with the theme being: **Learning and generation by random sampling**.

Finally, in the GitHub repository<sup>3</sup>, I have put notebooks of numerical applications illustrating the course since 2022. As much as possible, the notebooks can be run on Google Colab.

## 2. Lecture of January 17th

### 2.1 Introduction

The theme of this year **Learning and generation by random sampling** fits perfectly with the recent developments of the previous years, which have been marked by the release starting from 2022 of the conversational assistant ChatGPT<sup>4</sup> designed by the American

- 
1. <https://www.college-de-france.fr/chaire/stephane-mallat-sciences-des-donnees-chaire-statutaire/events>
  2. <https://www.di.ens.fr/~mallat/CoursCollege.html>
  3. [https://colab.research.google.com/github/jecampagne/cours\\_mallat\\_cdf](https://colab.research.google.com/github/jecampagne/cours_mallat_cdf)
  4. <https://openai.com/blog/chatgpt>

company OpenAI, specialized in the field of artificial intelligence. ChatGPT is built around a **language model** encoded in an architecture called **transformer**<sup>5</sup> with an enormous number of parameters ( $> 10^{18}$ ).

With GPT-3.5, the free version of ChatGPT, it responded to the request "*Can you make me a short list of what can be done with your version 3.5? You will phrase your answer in Latex using the itemize environment*" like this<sup>6</sup>:

- **Answering questions:** GPT-3.5 can provide answers to general questions based on the information it has learned during its training.
- **Generating creative text:** It can create stories, poems, scripts, and other forms of creative text based on the instructions you provide.
- **Writing assistance:** You can use GPT-3.5 to help you write articles, essays, reports, etc., by providing basic information.
- **Programming:** You can ask GPT-3.5 to generate source code for specific programming tasks by providing clear instructions.
- **Text translation:** It can help translate sentences or paragraphs from one language to another.
- **Conversation simulations:** GPT-3.5 can simulate conversations with fictional characters or even help practice dialogues for real situations.
- **Mathematical problem-solving:** You can ask it to solve mathematical problems or generate steps to solve them.
- **Idea generation:** It can be used to generate ideas for creative projects, marketing campaigns, etc.
- **Trivia questions:** You can ask it trivia questions, and it will attempt to provide answers based on the information it has learned.
- **Decision-making support:** GPT-3.5 can help explore different options and provide information to facilitate decision-making.

It is important to note that GPT-3.5 operates by generating responses based on

---

5. NDJE: architecture related to the attention mechanism, its implementation dates back to 2017: <https://arxiv.org/pdf/1706.03762.pdf>

6. NDJE: I removed the provided LaTeX code to give only the final rendering.

pre-trained language models and does not have real understanding or consciousness.  
It is essential to formulate clear instructions to obtain the desired results.

*NDJE:* Now, it is necessary to take a critical look at these functionalities as stated. In particular, when it "talks" about *Mathematical problem-solving*. Without going into details that would lead us astray, when asked the question "*Hello, do you have any useful notions about unipotent matrices, please?*", ChatGPT is very eager, but while trying to propose 5 properties, 3 of them were clearly false. From the "discussion" initiated to show it its errors, we observe that ChatGPT tries to correct them by reformulating the proofs, but ends up confusing its "neurons" like a student who hasn't digested the lesson well. Therefore, we see a limitation of the underlying model where the output may be revised by an expert. Many other experiments have confirmed these limitations. But it is undeniable that this free version has stirred up a lot of controversy. This has allowed everyone to use such an assistant and to discuss its theoretical, philosophical implications, even to develop organizational work schemes in all areas such as child education, student training, and applications in services and the industrial world. A revolution is underway.

The paid version (20\$/month) of GPT-4 (March 2023) offers even more stunning features such as<sup>7</sup>:

- **Interpreting** images and understanding their contexts;
- **Generating** images from a simple written description in natural language;
- **Generating** more creative texts (songs, movie scenarios...);
- **Summarizing** very long texts up to 25,000 words;
- **Creating** a website or an application from a simple sketch;
- **Generating** code to create a video game like Snake...

The term **generating** is omnipresent, and we can clearly see the link with this year's course. However, besides this ability to generate images, texts, the **interpretation** capability is equally astonishing. These two functionalities pose problems. Note that the possibility of generation allows for drifts well illustrated in the media. But in the context of the course, these language models challenge conventional wisdom. Not only can we question "how"

---

7. NDJE: the list is not generated by ChatGPT-3.5, which responds "*I would like to clarify that to my knowledge (with a last training update in January 2022), there is no GPT-4 version available...*

and "why" these models are so performant even if they can make mistakes, but they also challenge the way tasks are traditionally divided in statistical learning.

Indeed, it is customary to distinguish (one can refer to previous courses such as 2018, 2019 for example):

- **Supervised** learning where we have **data**  $x$ , and we want to find a **response**  $y$ : e.g., the class of an object for a **classification** task, or the value of a quantity for a **regression** task. Supervision lies in the fact that to perform a task, we have a training set  $\{x_i, y_i\}_{i \leq n}$  where we know the response  $y_i$  for each data  $x_i$ .
- **Unsupervised** learning for which the aim is to **build a model of the data**  $x$ . We will see that in this case, the question arises of modeling the **probability density**  $p(x)$  which allows for **generating** new data through **sampling**.

However, we observe that these two types of learning tend to be used complementarily, especially in the approach of transformers.

## 2.2 Digressions around two emblematic examples with GPT-4

S. Mallat gives us two examples of results from GPT-4 following his requests. The first example concerns an **image generation**. He submitted the request (the original was in French): *Image illustrating a course at the Collège de France on: "Learning and Generation by Random Sampling"*. The result is visible in Figure 1. It is clear that the image is of very good quality and above all that it has **a lot of indirect semantic content**. We clearly have a model of the world which, particularized by the request, gives according to this model a course at the Collège de France. We can see a sort of temple of knowledge with Greek statues, an amphitheater where we can distinguish a certain diversity of the audience with characteristics of people more or less old (e.g., hair color) which may differ from the actual audience. The male gender is clearly in the majority, which, however, reflects reality quite well. We note a language bias since the title of the course has been translated into English, which was not requested, furthermore, we clearly distinguish a blackboard with formulas/graphs which denotes a mathematics course, which was not directly suggested either. Finally, the person giving the course is a white man wearing glasses, which reflects the present situation quite well. Of course, we could further detail this image which is extremely rich in underlying information (note the diversity of the participants behind the speaker). How are such capabilities possible?

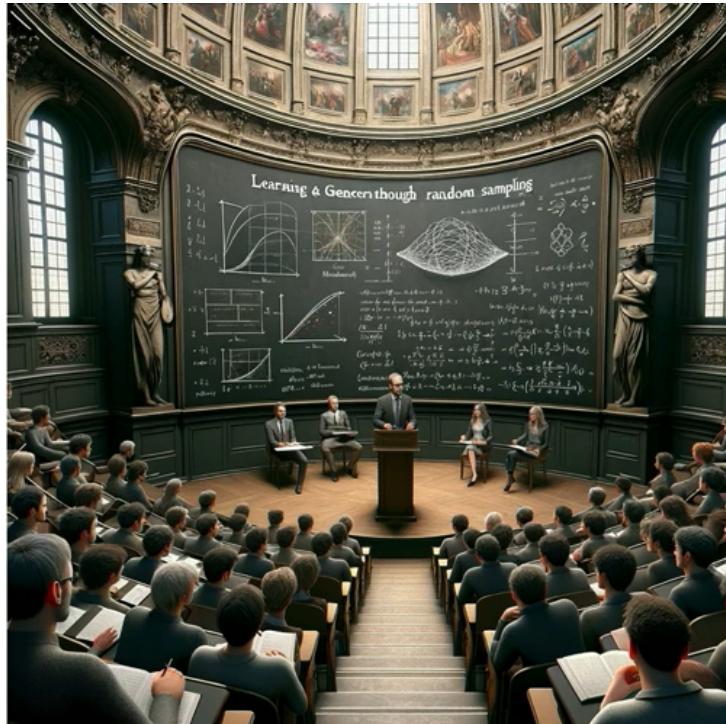


FIGURE 1 – Generation of an image by Chat GPT-4 from a simple request formulated in natural language. The image is accompanied by a short text explaining what the Collège de France is.



FIGURE 2 – Image for which ChatGPT-4 is asked: "Can you describe this image for me?".

The second example concerns the ability to **interpret the image** (Fig. 2) taken by a 16M pixel (8-bits RGB) mobile phone and uploaded to the site through the Dari interface, for which the simple natural language request is given (original was in French): "*Can you describe this image for me?*". The response given is as follows (*NDJE. except for transcription error(s), and the GPT response was in French too*):

The image shows a well-appointed room with wooden furniture. In the center, there is a turned-off television placed on a low cabinet with light-colored drawers. On each side of the cabinet, wooden cupboards are integrated into the wall, painted in yellow with orange and green borders. On the right, a bookshelf filled with books is against the wall, and there seems to be a door or a wardrobe adjacent to this bookshelf. The room appears to be illuminated by soft light, and you can see a piece of blanket or sheet, suggesting that the photo was taken from a bed.

Note that this type of task of describing an image is not new; it dates back to the early work on computer vision. Five to six years ago, S. Mallat tells us, for such requests, there were a few labels given as responses (e.g., "image of an apple"). Here, with GPT-4, we are clearly not playing in the same category, if we use a sports vocabulary. Not only is the formulated response grammatically correct, but what is impressive is the accuracy of

all the details mentioned, and the finesse of the analysis with the underlying context to formulate the last proposition.

**The trained language model is therefore capable of generating and interpreting, but also of answering all kinds of questions.** This is quite impressive even if there can be errors, and we understand the multiple questions that such capabilities raise, and the possibility of multiple uses. There is a **considerable gap** between, on one hand, what is produced by the **industry** which increasingly uses such systems that are being developed to be adapted to specific problems of this or that domain, and on the other hand, the **research** that is being done in parallel on algorithms and on understanding performance.

On the industrial side, to train such systems, it is necessary not only to acquire data, organize it, sometimes sort it, eliminate, select, etc., which requires a lot of human resources, but also, due to their gigantism, these networks can only be trained in specialized centers requiring very important material resources<sup>8</sup> with equally considerable operating costs. S. Mallat tells us that we don't even have the figures anymore because they become sensitive. For example, Gemini<sup>9</sup> from Google and GPT-4 are composed of approximately 1 trillion parameters, i.e.,  $10^{18}$  (Fig. 3). And Serge Escalè from the ITSocial website indicates that "the learning time of OpenAI is about 6 months. Thus the 10,000 Nvidia V100 GPUs (graphics processing units) used for computation consume 7,200 Mwh," which corresponds to an associated electrical cost, not to mention the environmental cost.

It should be noted that from an engineering (management) point of view, companies developing such architectures are reorganizing themselves to optimize their developments in order to ensure their sustainability. But this also entails a change in the way the performance of these models is disseminated: see for example the comparison rendering of Gemini Ultra and GPT-4 by D. Hassabis (CEO and co-founder of Google Deepmind) and S. Pichai (CEO of Google and Alphabet) on the site <https://blog.google/technology/ai/google-gemini-ai/#sundar-note>. How can we verify what is being claimed? Until now, S. Mallat tells us, algorithms were in the public domain, not out of openness but simply because **the value resided in the data**, so it was interesting to let the algorithms develop in the public domain. From now on, **the value is in the algorithms**. This is what happens in language models (conversational assistants) where we have access (paid for the most

---

8. NDJE: see <https://medium.com/riselab/ai-and-memory-wall-2cb4265cb0b8>

9. <https://deepmind.google/technologies/gemini>

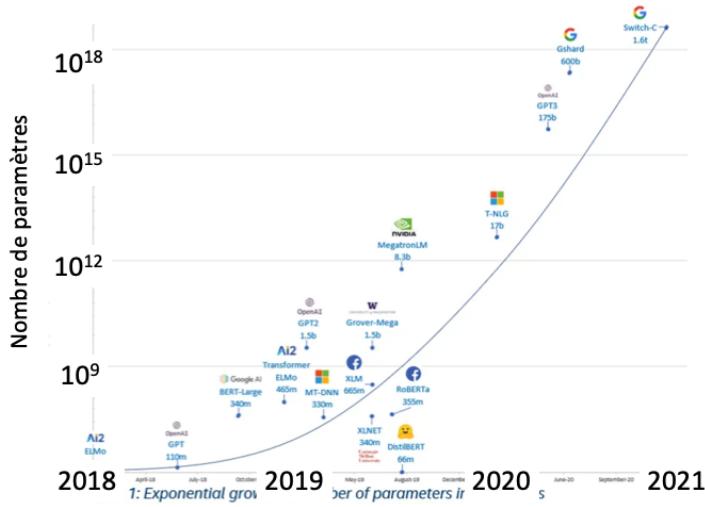


FIGURE 3 – Evolution of the number of parameters of AI architectures over time.

recent versions) to a simple interface. The case of OpenAI is somewhat exemplary for this company which, as its name indicates, initially had the intention of producing free software benefiting all humanity. After March 2019, it is a whole different story because the company must give dividends to shareholders. The game changes completely, but in a way it is a natural development (*business model*) of a company that needs capital to maintain its leadership and support competition.

All this still poses problems when conducting research on such language model architectures. Firstly, we generally do not have access to the same resources (financial, human, material), and secondly, imagine that a development allows you to obtain slightly better results for ImageNet-type image classification<sup>10</sup>, your result will go completely unnoticed, simply because the problem is much less difficult than the ones involved in the two aforementioned examples. So, **what is the relevance of "academic" research**, what is the impact of a researcher, how can he/she advance knowledge? These are important questions, and the good news is that there are many possible research directions.

For example, by addressing domains where the data resource is not as important as that used to train language models, and where algorithms must integrate more *a priori*

10. <https://www.image-net.org/>

information<sup>11</sup> and where the architectures are much smaller. Another research direction is in a way a transfer of know-how. Indeed, if transformer architectures are suitable for processing language, it is interesting to try them in other fields. For example, chemistry to produce new molecules, or pharmacology to obtain new drugs, materials science, fluid mechanics, physics of the infinitely small and infinitely large, and many others, all these fields benefit from the qualities of these new architectures.

Finally, there is a research domain that has been the subject of the lecture series since 2017, it is **trying to understand**. Where do emerging phenomena come from? ChatGPT/Gemini systems are trained in an unsupervised manner. Typically with texts collected on the Web, one can initially do training of the type "give me the missing word", then "the next sentence", etc. But currently, training is more complex with unsupervised phases and reinforcement phases, in order to control aspects related to "ethics" and/or "compliance with the law" for example. But understanding what happens in these architectures once trained is a difficult subject. S. Mallat mentions Benoit Sagot's lecture on language model engineering aspects<sup>12</sup>.

**This year's and next year's course is aimed at building the mathematics that help understand what happens when modeling for the purpose of generating new data.** Mathematics, S. Mallat tells us, are established over the long term, so patience.

*NDJE. Before continuing, I would like to address a topic that could of course be developed further. You may wonder at this point: what is the relevance of these lecture notes? Indeed, why not have GPT-4/Gemini watch S. Mallat's lecture and ask it to transcribe it, summarize it, clarify/illustrate a point, etc. I have no doubt that one day this will be done. The added value does not lie in simply transcribing the lecture. It is a rendering that, if it seems clear to me, will undoubtedly resonate with the reader. For example, you will sometimes see referenced articles, illustrations different from those presented, or digressions that tend to explain certain points that S. Mallat does not have time to address. Furthermore, having written the notes since 2017, I relate these lectures because S. Mallat cannot revisit what he has done in previous years every year, it would not make sense. The GitHub repository was implemented with the same intention of providing illustrations of concepts/algorithms in the form of Python code with light numerical experiments, related*

---

11. NDJE: see the 2020 Course on *A Priori Information in High Dimension*.

12. <https://www.college-de-france.fr/fr/chaire/benoit-sagot-informatique-et-sciences-numeriques-chaire-annuelle>.

*to the theoretical points of the course. It is clear that a model capable of answering all queries can exactly do the same work. But the parallel can be drawn between having all possible math books available and the added value of a teacher who, to develop his/her course, selects chosen excerpts with a certain bias, to create a coherent outline that can be understood by his/her audience. Moreover, the energy expenditure of a brain is 20 to 40 Watts which is nothing compared to the usage of current models.*

### 2.3 What is the mathematical framework?

We are working within a **probabilistic framework**, which has immediate consequences on the tools used. However, why choose such a framework? We can refer to the discussions in the 2022 Course Sec. 2.2 and the 2023 Course Sec. 3.1 and Sec. 3.2. In fact, in the case of supervised learning, the response  $y$  can be seen as the result of a deterministic function of the data ( $y = f(x)$ ), and where  $x \in \mathbb{R}^d$  can even be confined to a low-dimensional manifold which tends to simplify the problem. Therefore, the probabilistic framework is motivated by the choice of **modeling** above all (unsupervised learning). But in itself, this choice of modeling is not one for the following reasons.

First, we have data for which we want to establish a **model**, that is to say, naturally we are in the framework of **statistical estimation**, i.e., we want to calculate probabilities.

However, more fundamentally, the point that tilts the balance towards a probabilistic framework is that we are dealing with data of **very high dimension** by the nature of the data. We know that  $d$  is of the order of a few million for common images. However, we want to estimate probabilities of the type  $p(x)$  (or  $p(x|y)$ ), which is "impossible" in general, because of the sacred **curse of dimensionality**<sup>13</sup>. For memory, it translates into the fact that the number of necessary samples  $n$  grows exponentially with the dimension  $d$ . In order for the problem to be manageable, we need to inject *a priori* knowledge based on the fact that emergence must be the result of **underlying structuring**.

The ubiquitous notion that allows us to get out of this is **independence** (cf. Courses 2022 and 2023). Indeed, if a data point  $x$  is a vector of dimension  $d$ ,  $(x_1, \dots, x_d)$  and if

---

13. NDJE. See specifically the 2018 Course but it is a recurring theme in other courses as well.

each component  $x_j$  follows a distribution  $p_j$ , then

$$p(x) = p(x_1, \dots, x_d) = \prod_{j=1}^d p_j(x_j) \quad (1)$$

which means that we have replaced the problem of finding a function of  $d$  variables with the search for  $d$  functions of only one variable. Breaking down the difficult problem into  $d$  easier problems helps combat the curse of dimensionality.

That being said, this "total" independence is illusory in many practical cases: you can imagine, for example, the contours of an object in an image naturally relate pixels that do not necessarily have to be neighbors; or you are likely feeling the influence on our daily lives right now (e.g., oil prices) of decisions made by leaders of countries that are physically very distant from us. What to do in these cases? We can always write<sup>14</sup>

$$p(x) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdots = p(x_1) \prod_{j=2}^d p(x_j|x_1, \dots, x_{j-1}) \quad (2)$$

However, conditional probabilities become high-dimensional as  $j$  approaches  $d$ . The question is whether we can make assumptions about the dependence of  $x_j$  on all the variables  $(x_1, \dots, x_{j-1})$ ? Can't we invoke reduction? Probably, for example: pixels involved in the contours of an object are not dependent on all pixels in the image (and vice versa); similarly, the price of oil, in a first approximation, depends on decisions made by a few large oil-producing countries to turn the tap on or off, and on the volume of demand from consumer countries (admittedly more numerous). Thus, we feel that we can make assumptions about the level of interdependence of the variables  $x$ , or we can look for representations  $x \rightarrow x'$  such that the variables  $x'$  are less interactive than the original variables.

**The challenge in each problem is to understand the structures of conditional probabilities.** What is remarkable about the aforementioned large language models is that, in essence, they calculate nothing more or less than conditional probabilities. For example, to generate the image  $y$  in Figure 1 to satisfy the query  $x$  (the sentence submitted to GPT-4), it first calculated  $p(y|x)$  from  $p$  found during training, and then sampled the said conditional probability to give the response. Naturally, calculating the probability

---

14. NDJE. To simplify notation, we remove the indices from the probabilities.

$p(y|x)$  and sampling it in very high dimensions makes the task naturally very complex. So, it is necessary to understand the mechanism that allows practical realization and ask the question: what are the structures at different scales (cf. Course 2019 Sec. 3.7) of conditional probabilities that allow combating the curse of dimensionality.

Finally, an important point that will be discussed in the course concerns **the computational complexity in high dimensions**. Indeed, to perform calculations of means, marginal probabilities, etc., we need to carry out **integrals** in high dimensions. A priori, the number of operations must grow exponentially with the dimension  $d$ , unless we use random methods called **Monte Carlo**<sup>15</sup>. These techniques involve not computing these integrals on grids of the space  $\mathbb{R}^d$  but rather randomly sampling and summing/averaging them. We will see why we avoid the curse of dimensionality, although it remains true that generation can be difficult in practical cases in high dimensions (e.g., the case of multi-modal distributions).

Thus, we have reviewed **why the probabilistic framework is central and directly related to the fact that we are constrained to operate in high dimensions**.

## 2.4 Link between supervised and unsupervised training

The link in question is not new as it arises from the **Bayesian perspective**, as we will see.

Regarding *supervised* learning where we are interested in obtaining the response  $y$  from  $x$ , in the probabilistic framework, we then desire to obtain  $p(y|x)$ . If we are interested in a classification problem, the Bayes classifier<sup>16</sup> is the one that on average produces the minimum error, it maximizes  $p(y|x)$ :

$$\hat{y} = \operatorname{argmax}_y p(y|x) \quad (3)$$

On the other hand, regarding *unsupervised* learning in the probabilistic framework, we are rather interested in modeling the probability  $p(x)$ . But there is a link with the

---

15. NDJE. In 2023, I developed examples of such techniques in the Python notebook `Monte_Carlo_Sampling`. The notebook is accessible here: [https://github.com/jecampagne/cours\\_mallat\\_cdf/blob/main/cours2023/Monte\\_Carlo\\_Sampling.ipynb](https://github.com/jecampagne/cours_mallat_cdf/blob/main/cours2023/Monte_Carlo_Sampling.ipynb).

16. NDJE. see e.g., Course 2022 Sec. 2.2 or/and Course 2023 Sec. 3.4.2

previous because we have Bayes' formula<sup>17</sup>:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} \quad (4)$$

in which we recognize  $p(y|x)$  the *posterior* probability,  $p(x|y)$  the likelihood,  $p(y)$  the *prior* probability (e.g., the prior probability of being in one class rather than another), and  $p(x)$  the evidence. If we want to maximize  $p(y|x)$  according to  $y$  then

$$\hat{y} = \operatorname{argmax}_y p(x|y)p(y) \quad (5)$$

but  $p(y)$  is rather to be seen as data, and therefore in this problem the practical and delicate point is to maximize the likelihood  $p(x|y)$ .

Schematically, one can then think of proceeding according to the following sequence of operations: take the data  $x$ , group them into different classes  $y$ , and **model each of the classes**. That is to say, in the end, we could be able to generate new elements from the different classes. But, in practice, for some problems, it is not necessary to do this work and we can use another approach. Think of the case of classifying an image into two categories depending on whether it contains a fire truck or a black car. Modeling all images that contain either a fire truck or a black car is a priori extremely complex given the diversity of situations in which these vehicles operate. In contrast, it is very likely that simply counting the number of red and black pixels is sufficient to provide a good indicator of the image class.

In fact, **extracting discriminative elements (features)** for a given problem is a first approach that can be very effective and naturally contributes to dimensionality reduction. It is clear that the modeling approach was not at all the one implemented a few years ago, and so there were two different philosophies/approaches. In a way, computing  $p(x|y)$  to distinguish between two classes can be considered a very brute force approach. However, in the case of an image, besides knowing if it contains a fire truck or a black car, one can potentially ask many other questions, so the modeling approach becomes much more effective.

It is the necessity to **answer all sorts of questions that imposes the modeling of**

---

17. NDJE. Reminder:  $p(x, y)$  the joint probability is equal to the products  $p(y|x)p(x)$  and  $p(x|y)p(y)$ .

$p(x)$ ,  $p(y|x)$ . That's why the two learning domains tend to converge. **We are currently tackling complex domains where shortcuts, even clever ones, like implementing low-dimensional representations, are no longer effective.** However, it is important to keep in mind that not all problems are of the same nature, and in many cases, for example, those where obtaining a large number of data points is difficult, these low-dimensional representations are very useful.

## 2.5 What are the issues?

We can mention three types of problems that will interest us, starting with **modeling**. Recall that we need to model  $p(x)$  or  $p(y|x)$ . The use of **parameterized families**  $\{p_\theta(x)\}_\theta$  is used quite naturally<sup>18</sup>.

However, we need to choose a family of functions that is well-suited. This is an **approximation problem**, because we must be able to find the best  $\theta$ , denoted  $\theta^*$ , and ensure that  $p_{\theta^*}$  is a good approximation of the actual distribution  $p(x)$  (which is assumed to exist *a priori*). The metric used during training is the Kullback-Leibler "distance", defined as follows<sup>19</sup>:

$$D_{KL}(p||p_\theta) = \int p(x) \log \frac{p(x)}{p_\theta(x)} dx \geq 0 \quad (6)$$

(reminder: if it is zero then  $p = p_\theta$ ). To find  $\theta^*$ , we need to minimize  $D_{KL}(p||p_\theta)$  over all values of  $\theta$ , so it is an **optimization problem**. More precisely, it is a **maximum likelihood problem**, which is fundamental in statistical estimation introduced by **Ronald A. Fisher** in 1922 (see e.g., Course 2022 Secs. 2.1 and 2.3).

Finally, we can use this modeling and approximation/optimization scheme to perform **inference**. For example, in the medical field, we typically have symptoms (the data  $x$ ) such as cough, nausea, fever, vomiting, diarrhea, difficulty breathing, heart palpitations, etc., and we want to provide a response ( $y$ ) in terms of diagnosis such as COVID, flu, pneumonia, tachycardia, any type of failure, etc. In this context, we may have data-

---

18. NDJE: See the 2022 Course with R. Fisher's approach

19. NDJE. See also the 2019 Course Sec. 7.2.3, for example.

bases, such as the Quick Medical Reference<sup>20</sup>, which describes *573 diagnoses, recognizes 4,100 patient observations, and includes over 4,000 links detailing causal, temporal, and probable relationships between disorders*. A doctor, of course, wants to be able to provide a diagnosis based on a few symptoms of their patient. Let  $(x_1, x_2)$  be the symptoms, we need to be able to calculate  $p(y|x_1, x_2)$  and, for example, select the  $y$  with the maximum probability. Now, if we have the conditional probability  $p(y|x_1, x_2, \dots, x_d)$  then we need to perform the (marginal) integral<sup>21</sup>:

$$p(y|x_1, x_2) = \int p(y|x_1, x_2, \dots, x_d) \prod_{j=3}^d dx_j \quad (7)$$

Therefore, making an inference from the model will require performing integrals (in high dimensions) of conditional probabilities, which brings us to the issues of Monte Carlo sampling. In doing so, mentioning Monte Carlo implies performing **sampling**, and therefore being able to do **generation**.

## 2.6 Panorama of the Mathematical Domain

### 2.6.1 Modeling

When tackling the problems mentioned in the previous sections, there are many different concepts, and we will take the time to address them. Initially, the **Monte Carlo** techniques will occupy us. It's certainly not necessarily obvious, but we will see that it can be done well. However, what is more complicated is the **modeling**.

However, it is clear that modeling  $p(x)$  in 1D is much easier. We can view this as an interpolation problem. In this context, the 2021 Course provides some entry points. For example, throughout the field of low-dimensional analysis, orthogonal bases are chosen according to the regularities of the function  $p(x)$  to be approximated. The vector  $\Theta$  consists of the coefficients in the chosen basis. Where the problem becomes complicated is when  $x \in \mathbb{R}^d$ . Indeed, trying to consider functions in the space  $L^2(\mathbb{R}^d)$ , we come back to the curse of dimensionality, namely that the number of basis elements becomes far too

---

20. <https://www.nlm.nih.gov/research/umls/sourcereleasedocs/current/QMR/index.html>

21. NDJE. for simplicity, we have ordered the variables  $(x_i)_i$ .

large. We need to proceed differently, and for that, we will use a tool already implemented especially in the 2023 course, namely **Gibbs probabilities**:

$$p_{\Theta}(x) = Z_{\Theta}^{-1} \exp \left\{ - \sum_k \theta_k \phi_k(x) \right\} = Z_{\Theta}^{-1} e^{-\Theta^T \Phi(x)} = Z_{\Theta}^{-1} e^{-U_{\theta}(x)} \quad (8)$$

with  $\Phi(x) = (\phi_k(x))_{k \leq K}$ . The constant  $Z_{\Theta}$ , *the Gibbs partition function*, ensures the normalization of the probability. By analogy with Statistical Physics, we denote  $U_{\theta}(x) = \Theta^T \Phi(x)$  the **internal energy** and  $F_{\theta} = -\log Z_{\theta}$  the **free energy**. **The underlying idea is that through the functions  $\phi_k$  (and thus  $U_{\theta}$ ), we can inject a lot of *a priori* information that accounts for structures.**

In this framework, **Markov fields** will be an important subject of study, as with these, we can conceive a first type of models that allow us to understand modeling in high dimensions. In fact, with Markov fields, we favor **local** interactions, which breaks the curse of dimensionality. We will see the **Hammersley-Clifford theorem**, which gives the necessary and sufficient conditions under which a strictly positive probability distribution can be represented as events generated by a Markov field. This theorem links conditional independence and the way to write  $p(x)$  in the form of a Gibbs distribution, with a particular expression that reflects this independence. However, this locality assumption is not sufficient in itself because we have previously mentioned examples where clearly non-local interactions are not negligible. However, we will try to find transformations that will make this locality assumption more viable.

Beyond representations that place us in a framework of local interactions, there is the notion of **multi-scale** problems that is important (Fig. 4). Underlying this is the assumption of a **hierarchical organization** of the world. This is somewhat true in Physics when we look at structures on small, medium, and large scales, i.e., from Elementary Particle Physics to Cosmology, passing through the intermediate scales of life. There is a manifest organization at all scales, and to some extent, we can study each scale independently, at least in a first approximation. However, this hierarchy can also serve as an approximation of the evolution of agents in a social network, for example. The fundamental concept that emerged from Physics is the **Renormalization Group**<sup>22</sup>. Note

---

22. NDJE. We will not cover this area in this year's course.

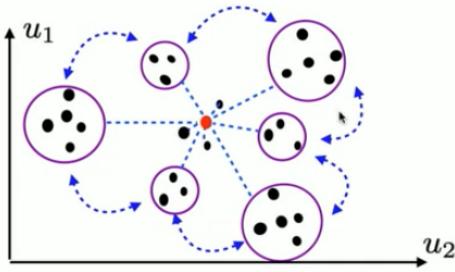


FIGURE 4 – Schematization of local interactions between the red agent and its neighbors, and the consideration of interactions on larger scales by grouping agents. This allows for modeling in  $O(\log d)$  terms.

that we will not address this domain in this year's course.

If we want to further complexify the modeling of  $U(x)$ , we can use **neural networks**. An incidental remark: neural networks fit into the modeling approach. In this regard, neural networks are not "separate" as intruders. And if they are effective, then the ideas presented above must emerge in one way or another. In fact, it is the **architecture** that carries concepts like **hierarchy** (e.g., U-Net architecture), **local interactions** through small-dimensional convolutional filters, and **non-linearities** that reinforce the complexity of the modeling.

Finally, an additional idea that has enabled the development of large language models lies in **transformers** with the notion of **attention**. When considering Markov fields, we look at local interactions. However, it is known that in some cases, this is not sufficient because distant information can propagate and lead to interactions of the same order of magnitude (or even larger). Other examples than those already mentioned to illustrate the point: a small crack in a material can propagate and break the structure; a negation in a sentence can change the meaning of an entire paragraph (the same could be true with the placement of a comma), etc. Therefore, in order to capture such phenomena, it is necessary to look beyond neighbors. This is actually quite natural. S. Mallat describes how the brain provides an image of a scene from the constant and imperceptible movements of the eyes: we constantly make rapid movements of our eyes that allow important elements of our environment to be placed within the fovea, which perceives the details of a scene finely but is very small (1.5mm in diameter at the center of the macula). These

rapid movements are not random; they are **points of attention** where there is important information to understand the structure of the scene. Therefore, the notion of attention has always been at the heart of cognitive psychology; what is new is that a way has been found to implement it and make it effective. These transformers have allowed for a sort of **generalization of the notion of neighborhood** which is no longer *a priori* (as in setting a grid and a distance) but **learned**.

## 2.6.2 Sampling

In parallel with the modeling of probabilities, we have seen in the previous sections how important it is to be able to generate new data through **sampling**.

Just as we feel that methods for modeling  $p(x)$  in 1D are well understood, sampling in 1D, while not obvious, is not difficult. Initially, we focus on creating a **uniform random variable** from a **chaotic system**, which involves the notion of **ergodicity** already discussed during the 2023 Course (e.g., Sec. 6.2). Then, it is necessary to transform the uniform distribution to obtain the desired distribution. In this case, there are methods and algorithms such as *inverse transform sampling*, rejection-acceptance (*hit and miss*), or *importance sampling*<sup>23</sup>. This step will allow us to introduce important concepts in the course.

That said, we need to address the challenging case of high dimensionality. The central tool we will use, also covered in the 2023 course (e.g., Sec. 6.4 and Sec. 7), consists of **Markov chains**. The idea is to start from a probability distribution  $p_0$  that is easy to generate (e.g., uniform distribution, Gaussian, ...), then to apply a **transport** ( $T$ ) that iteratively should lead us to the target distribution. Thus, starting from  $x_0$  a sample from  $p_0$ , we iterate to obtain  $x_t = T^t(x_0)$ , and if we have chosen  $T$  well, then  $x_\infty$  is a sample from  $p$ .

In fact, if **the distribution  $p$  is an invariant** (fixed point<sup>24</sup>) of the transport and if convergence is fast, then generating samples according to  $p(x)$  is achievable in practice. We will study a number of algorithms (**Metropolis**, **Metropolis-Hastings**<sup>25</sup>, and Gibbs

---

23. NDJE: see the notebook [https://github.com/jecampagne/cours\\_mallat\\_cdf/blob/main/cours2023/Monte\\_Carlo\\_Sampling.ipynb](https://github.com/jecampagne/cours_mallat_cdf/blob/main/cours2023/Monte_Carlo_Sampling.ipynb).

24. NDJE. See Course 2023 Sec. 6.5.

25. NDJE. See the notebook of the footnote 23.

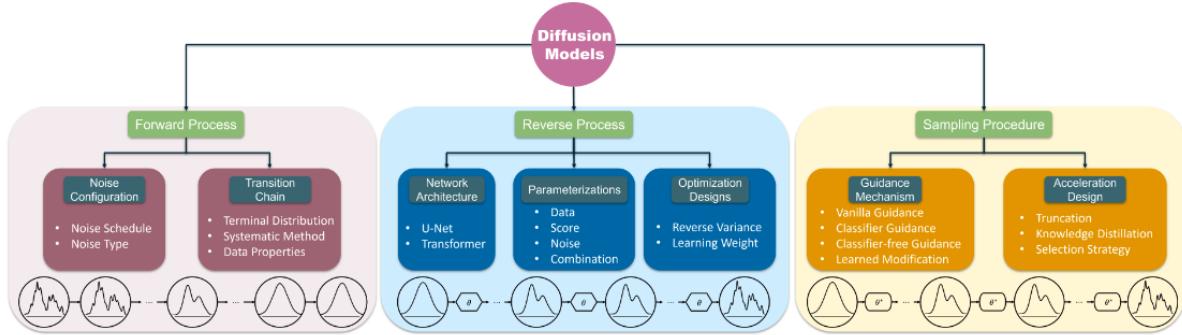


Fig. 1. The overview of diffusion models. The forward process, the reverse process, and the sampling procedure are the three core components of diffusion models, which are responsible for adding noise, training networks, and generating samples, respectively.

FIGURE 5 – Overview of diffusion models. The forward process, the inverse process, and the sampling procedure are the three main components of diffusion models, which are responsible for adding noise, training networks, and generating samples, respectively (*source arxiv:2306.04542*).

sampling,...) and their convergence. To do this, it is easier to view  $t$  as a *continuous time*. This transition from discrete to continuous allows us to use many more mathematical tools. We will thus obtain **stochastic differential equations**, including those of Paul Langevin (1872-1946) and Adriaan Fokker (1887-1972) and Max Planck (1858-1947), both developed to understand Brownian motion.

These equations are somewhat gradient descents of the energy ( $\nabla_x U(x)$ ) with added noise. The analysis then yields very nice results where convergence is characterized by "log-Sobolev" constants. The mathematical object that emerges from this analysis is the **Hessian**<sup>26</sup> of  $U$ . This is not surprising, indeed recall that we seek to minimize  $U_\theta(x)$  (increase  $p(x)$ , cf. maximum likelihood), hence the gradient descent whose convergence is all the faster as the Hessian is well-conditioned<sup>27</sup>.

To get closer to large generative models, there are "**Score Diffusion**" algorithms<sup>28</sup>.

26. NDJE. see e.g., Course 2022 Sec. 3.6.2 and Sec. 4.2.

27. NDJE. See the theorem (Th. 4) discussed in Course 2022 Sec. 3.6.2

28. NDJE: Here are two references for interested readers: J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in International Conference on Machine Learning. PMLR, 2015, pp. 2256–2265 (arXiv:1503.03585), and Chang, Ziyi;

In short, this sampling technique is an adaptation of Langevin's method by means of a homotopic method, i.e., one continuously evolves an initial distribution towards a final distribution (Fig. 5). For example, one starts with an image containing a lot of noise that is gradually reduced. And instead of minimizing the Kullback-Leibler divergence, one optimizes the **Fisher Information** (Course 2022 Sec. 5.3) using the "**score matching**" method<sup>29</sup>. It turns out that this method is much more efficient and faster but it only works if  $U_\theta$  satisfies certain conditions.

## 2.7 Some illustrations

*NDJE. At this stage of the course S. Mallat presented us with some results using the video-projector. For some of them I refer you to the video put on line on the Collège de France (after 1:04:18).*

First of all, S. Mallat lets us hear sound frames, or see images of turbulence making the link with the 2023 Course Sec. 2.9. This illustrates the fact that Gaussian models ( $U_\theta = \frac{1}{2}x^T C^{-1}x$ ) are not capable of capturing structures (in audio it is the intermittency). Having said that, it is not easy to go beyond these simple models. S. Mallat tells us that 30 years ago all modelling was basic Gaussian and the question was to study small deviations.

Once we have understood how to model non-local interactions with correlations between scales, we can, for example, generate images of non-Gaussian fields as shown in figure 6. Scattering networks of order 2 can also be implemented (Course 2023 Sec. 9.9). However, S. Mallat tells us that he looks at these results, in which he took part, with a certain air of amusement in the face of the results of generative models of the GPT-4 type. But he tells us that there is a need to really understand what is going on, and it's not a question of generating pretty pictures. This process requires mathematics and starting with simple (but not simplistic) examples.

Going further, as mentioned in the previous section, involves modelling with diffusion models, using a stochastic differential equation (Langevin) that we will invert<sup>30</sup>. Initially

---

Koulieris, George Alex; Shum, Hubert P. H. (2023). "On the Design Fundamentals of Diffusion Models: A Survey". arXiv:2306.04542

29. NDJE: where the score is  $\nabla_\theta \log p_\theta(x)$ .

30. NDJE: see for example Yang Song et al. (2021) arXiv:2011.13456.

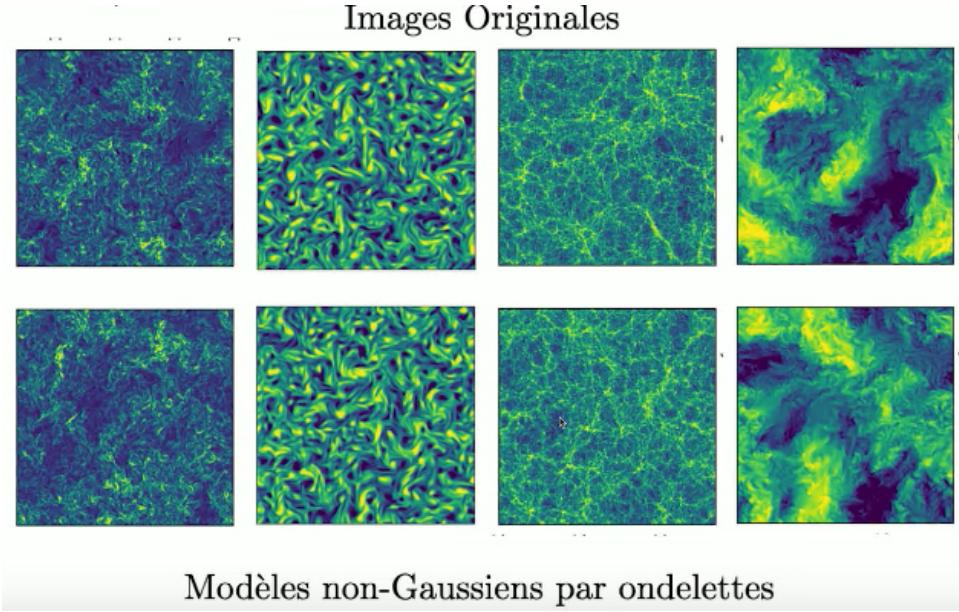


FIGURE 6 – Example of generation of non-Gaussian fields by wavelet models: *turbulence*, *cosmic web*, etc. (source [arxiv:2306.17210](https://arxiv.org/abs/2306.17210))

(Fig. 7), we start with an image of a face without noise, then progressively through the evolution (forward) of the Ornstein-Uhlenbeck equation<sup>31</sup>, the noise is injected up to the point where only noise is visible. Then the process is reversed (backward) and the original image is restored. Once the expression for the energy  $U(x)$  is known, when generating an image (backward phase) from (white) noise, the term  $\nabla_x U(x) = -\nabla_x \log p(x)$  is there naturally to maximise the probability, and to give typical examples of  $p(x)$ . If mathematics seem clear and established since around the 1930s, the point emphasized by S. Mallat is that the estimation of the score was once thought to be impossible. However, the situation has changed with neural networks, which make it possible to compute the gradient of  $U(x)$ . The network optimization is done through "score matching".

Note that for each new noise image  $x_0$ , there corresponds a new image generated from  $p(x)$  after the denoising process, allowing for the generation of new portraits of "people who do not exist," as well as scenes of landscapes, interiors of houses, etc. But the legitimate question arises: if we have a database of faces, then train a neural network using

31. Leonard Salomon Ornstein (1880-1941) and Eugene Uhlenbeck (1900-88), both physicists

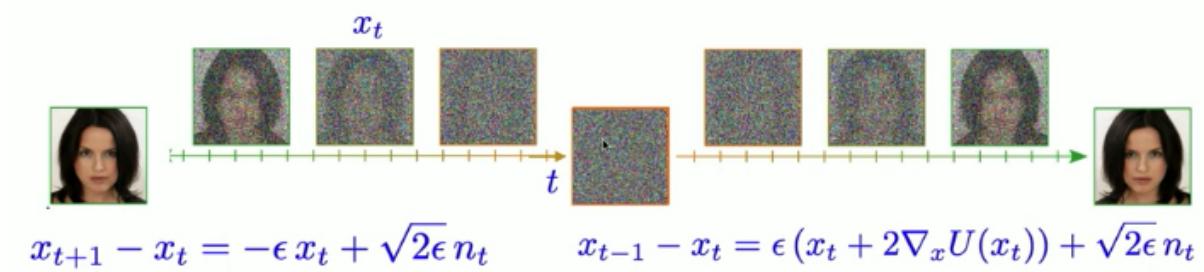


FIGURE 7 – Example of the evolution of an image by the stochastic process of a diffusion model: on the left, a progression where noise is injected (forward) and on the right, where denoising is carried out

score matching, and synthesize new faces through backward diffusion, **do we ultimately obtain for each new face a kind of mixture of all the faces in the database?** What happens if the training database is divided into two sets, S1 and S2, the sizes of which vary, and after training two models, new faces are generated from the same noise image? **The underlying question is that of generalization.**

The answer to this question may lie in a recent article by Kadkhodaie et al. from 2023, to which S. Mallat contributed<sup>32</sup>, which is an example of ongoing research. Figure 8 tends to show that with small training databases ( $N \leq 10$ ), the models only "memorize" the database and are unable to produce new faces (overfitting). If the generation process is initialized with the same noise image, the final images produced by the two models display faces from their respective databases and are naturally different by construction. For databases containing  $O(100)$  different faces, the models start to produce new "faces," but these do not correspond to the reality of a well-formed face. It is only from databases with sizes of  $O(10^4)$  that the models generate new faces that conform to reality. Furthermore, even more strikingly, models trained with the S1 and S2 sets of this size, which, we recall, have no images in common, each produce a new face that turns out to be almost identical if the two models are initialized with the same noise image. This is truly generalization. **The interpretation is that the transport of white noise to the final image learned by the two**

<sup>32</sup> Kadkhodaie, Z., Guth, F., Simoncelli, E. P., Mallat, S. (2023). *Generalization in diffusion models arises from geometry-adaptive harmonic representation*. arXiv.2310.02557

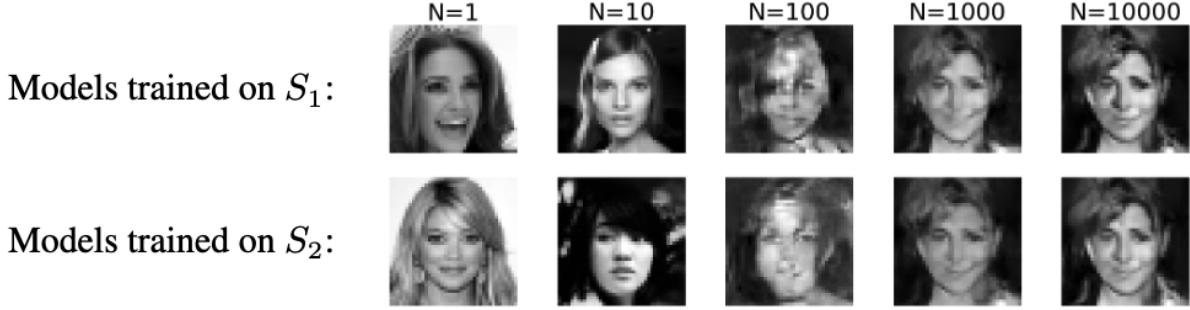


FIGURE 8 – Diffusion models are trained on distinct subsets  $S_1$  and  $S_2$  of a face dataset. The subset sizes  $N$  range from 1 to  $10^4$ . Then, a sample from each model is generated using deterministic inverse diffusion initialized with the same noise image. For training set sizes  $N = 1$  and  $N = 10$ , the networks memorize, producing samples drawn from the training set. For  $N = 100$ , the generated samples do not match any image in the training set but are heavily corrupted. This corresponds to a regime where the networks transition from memorization to generalization. For  $N = 10^4$ , the two networks generate almost identical images.

**independent models is the same, and it is the same distribution  $p(x)$  (or equivalently  $U(x)$ ) that is learned, once a sufficient number of training images are available.** There is a kind of smooth transition from a state where the models are simple "parrots" to a state where they become autonomous "speakers".

*NDJE: I allow myself a small addition. It is not uncommon in supervised learning to divide the dataset into multiple sets: for example, a training set, a test set to tune model parameters, and finally a validation set to ultimately derive top-5 accuracy values in the case of a classification task, for example. Moreover, one can have multiple models and disjoint training sets in the same line to ultimately compare performances. One can even use two models of identical architectures but with different parameter initializations. In this context, it is common and appropriate to find that the performances of two models are statistically identical; otherwise, one invokes bias, systematic error, etc. Kadkhodaie et al.'s result is of a different nature as it is within the framework of unsupervised learning.*

The important consequence, contrary to the initial criticisms expressed by "experts,"

S. Mallat tells us, is that **GPT-4-like language models are not just large memories** that would simply amalgamate different elements of the database during new queries. There is something much deeper.

## 2.8 Course Outline

To conclude this presentation session on this year's theme, S. Mallat quickly outlines the course plan:

- We will start with Monte Carlo methods.
- We will then review modeling, learning, and sampling concepts in a general manner.
- Next, we will delve into modeling using Markov fields,
- followed by sampling, first in 1D, then via Markov chains.
- To delve deeper, after transitioning to continuous time, we will study the Langevin equation.
- Then, we will address the theme of "score." However, if time is lacking, this will be postponed until next year.

Reminder: seminars are important to shed light on current practices.

## 3. Lecture of January 24th

During this session, we revisit the motivations for using random models, then we delve into Monte Carlo methods for high-dimensional calculations. Finally, we address the triptych of "Modeling, Learning, Inference."

### 3.1 Random Models: Why?

We are considering, as illustrated in the last session, high dimensionality. But let us recall that it is not obvious *a priori* that we should adopt a probabilistic framework. Let's consider the example of Gaussian white noise<sup>33</sup>.

---

33. NDJE. See also Course 2023 Sec.2.6

From a probabilistic standpoint, Gaussian white noise is a probability distribution of  $d$  independent variables, each of which is distributed according to a Gaussian distribution, for example,  $x_i \sim \mathcal{N}(0, \sigma^2/d)$ . Thus,

$$p(x_1, x_2, \dots, x_d) = \prod_{i=1}^d p_i(x_i) = \prod_{i=1}^d \exp\left\{-\frac{x_i^2 d}{2\sigma^2}\right\} = \exp\left\{-\frac{\sum_{i=1}^d x_i^2 d}{2\sigma^2}\right\} = \exp\left\{-\frac{\|x\|^2 d}{2\sigma^2}\right\} \quad (9)$$

Now,  $z = \sum_{i=1}^d x_i^2$  is a random variable. If the  $x_i$  are *iid* according to  $\mathcal{N}(0, 1)$ , we obtain that  $p(z)$  is the distribution of the  $\chi^2$  with  $d$  degrees of freedom<sup>34</sup> whose expectation is  $\mathbb{E}[z] = d$ . By change of variables, we then obtain

$$\|x\|^2 = \sum_{i=1}^d x_i^2 \xrightarrow[d \rightarrow \infty]{} \sigma^2 \quad (10)$$

and similarly, the variance of the  $\chi^2$  with  $d$  degrees is  $2d$ , thus

$$Var(\|x\|^2) \xrightarrow[d \rightarrow \infty]{} \frac{2\sigma^4}{d} \quad (11)$$

Thus, the localization of samples of Gaussian white noise occurs on a spherical shell whose thickness tends to 0 as  $d$  increases. Therefore, in high dimensions, this localization can be seen as a spherical manifold (Fig. 9). This is quite general; **in high dimensions, thanks to the independence property of variables**, we witness **concentration phenomena on manifolds**. We could therefore adopt a **purely deterministic perspective**, which is essentially the usual perception we have of the world. Note that during the 2023 course, we demonstrated how, in statistical physics, macroscopic phenomena emerge due to the law of large numbers.

However, this deterministic view would lead us to perform geometry (on manifolds) in very high dimensions. However, this is very complicated. Let's take the case of the sphere in Figure 9. Imagine a unit vector  $v$  emanating from the center of the sphere, and let's take a point  $x$  on this sphere. Let's examine the dot product  $\langle v, x \rangle$ :

$$z = \langle v, x \rangle = \sum_{i=1}^d x_i v_i \quad (12)$$

---

34. NDJE. The  $\chi^2(d)$  distribution is  $(1/2)^{d/2}/\Gamma(d/2)z^{d/2-1}e^{-z/2}$ .

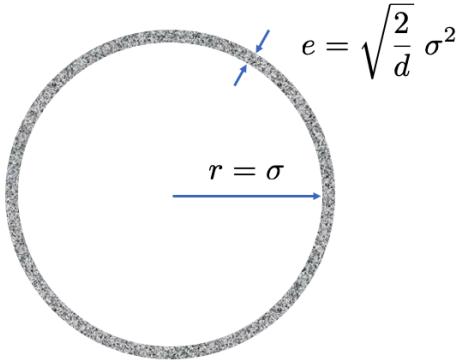


FIGURE 9 – Localization of  $\|x\|^2$  if all components are iid  $x_i \sim \mathcal{N}(0, \sigma^2/d)$  in dimension  $d$ .

Since the  $x_i$  are iid Gaussian random variables, it follows that  $z$  is also a Gaussian random variable, namely  $\mathcal{N}(0, \sigma^2/d)$ . So, when  $d$  is large, not only is the mean zero, but the variance of the dot product  $\langle v, x \rangle$  also tends to 0. Thus, the  $x$  are as if they were **concentrated on the equator** perpendicular to the vector  $v$ . Thus, these concentration phenomena are not necessarily intuitive.

Therefore, it's not so much that the deterministic framework would be unfeasible, but rather it would be very complex to implement. In the last session, we discussed the reasons related to the very high dimension that lead us to adopt the probabilistic framework. For example, the need to compute integrals is only made practicable by using the Monte Carlo method(s).

### 3.2 The Monte Carlo Method

This technique is not very old; its beginnings can be traced back to the 1940s during the nuclear bomb development program at Los Alamos. The initial problem was described by a deterministic scheme: counting the number of scatterings of a neutron before it collides with an atom nucleus in a reactor core and estimating the energy lost in the collision. Similar deterministic schemes can be used in game theory, where the rules are deterministic. It's somewhat a counting problem. However, the practical problem that interested atomic physicists was impossible to solve using these classical techniques.

*NDJE. The simulation methods, now called Bayesian in Particle Physics & Cosmology in particular, are very natural. Take for example the interactions of the neutron or any other type of particles: we draw the probability of occurrence of such or such phenomenon as we progress through the different materials encountered by the particle. These techniques are of course at work, for example, to study the progression of secondary particles during collisions of proton beams in the CERN LHC in the detectors Atlas, CMS, LHCb and Alice.*

Following the ideas of Stanisław Marcin Ulam (1909-84) and John von Neumann (1903-57) at Los Alamos, there were articles, notably those of Nicholas Metropolis (1915-99) around Markov chains that allowed for random simulations such as the Monte Carlo Markov Chain (MCMC). In this way, one can calculate expectations by simulation. Recall, for instance, the Buffon's needle experiment<sup>35</sup> in 1733, which also computed the decimals of the number  $\pi$  by an expectation calculation performed by simulation.

In fact, computing an expectation amounts to calculating an integral:

$$\mathbb{E}_p[X] = \int x p(x) dx = \mu \quad (13)$$

Now, if we sample  $p(x)$  by producing  $n$  values  $(x_i)_{i \leq n}$ , then we can define an estimator of the mean:

$$\hat{\mu}_n = \frac{1}{n} \sum_{i=1}^n x_i \quad (14)$$

It is unbiased, meaning  $\mathbb{E}_p[\hat{\mu}_n]$  is equal to  $\mu$ . Moreover, if the random variables  $(x_i)_i$  are iid with finite variance  $\sigma^2$ , then the variance of  $\hat{\mu}_n$  is given by

$$\sigma^2(\hat{\mu}_n) = \frac{\sigma^2}{n} \xrightarrow{n \rightarrow \infty} 0 \quad (15)$$

Finally, the law of large numbers<sup>36</sup> tells us that

$$\mathbb{P}\left[\hat{\mu}_n \xrightarrow{n \rightarrow \infty} \mathbb{E}_p[X]\right] = 1 \quad (16)$$

35. Georges-Louis Leclerc, comte de Buffon (1707-88).

36. NDJE. Course 2022 Sec. 3.3

As the random variables  $(x_i)_i$  are iid, we have a little more via the following theorem:

**Theorem 1 (Central Limit Theorem)**

$$\sqrt{n}(\hat{\mu}_n - \mathbb{E}_p[X]) \xrightarrow[n \rightarrow \infty]{\text{cv. in law}} \mathcal{N}(0, \sigma^2) \quad (17)$$

which means

$$\lim_{n \rightarrow \infty} \mathbb{P}\left(\sqrt{n}(\hat{\mu}_n - \mathbb{E}_p[X]) \leq z\right) = \Phi(z/\sigma^2) \quad (18)$$

where  $\Phi(z)$  is the cumulative distribution function of the normal distribution:

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_0^z e^{-x^2/2} dx = \frac{1}{2} \left(1 + \operatorname{erf}\left(\frac{z}{\sqrt{2}}\right)\right) \quad (19)$$

with  $\operatorname{erf}$  being the Gauss error function<sup>a</sup>. Therefore, we can obtain asymptotic confidence intervals. In particular,

$$|\mu_n - \mathbb{E}_p[X]| \simeq \frac{\sigma}{\sqrt{n}} \quad (20)$$

---

a. Milton Abramowitz and Irene Stegun, Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, section "Error Function and Fresnel Integrals"

Why does this interest us? In fact, we can always see an integral as follows:

$$\int_{\mathcal{D}} f(x) dx = \int_{\mathbb{R}^d} \mathbf{1}_{\mathcal{D}}(x) f(x) dx \quad (21)$$

where  $\mathbf{1}_{\mathcal{D}}$  is the indicator function of the set  $\mathcal{D}$ . Now, we can always define a **uniform** probability density on  $\mathcal{D}$  as follows<sup>37</sup>:

$$p(x) = \frac{\mathbf{1}_{\mathcal{D}}(x)}{|\mathcal{D}|} \quad (22)$$

---

37. NDJE. nb. It is customary to denote the volume of a set  $\mathcal{A}$  as:  $|\mathcal{A}|$ .

Thus,

$$\int_{\mathcal{D}} f(x) dx = |\mathcal{D}| \int_{\mathbb{R}^d} f(x) p(x) dx = |\mathcal{D}| \times \mathbb{E}_p[f[X]] \quad (23)$$

which gives us access to **the value of the integral via the calculation of an expectation**. Let  $U$  be a *uniformly distributed random variable* on  $\mathcal{D}$ , then by drawing  $n$  samples  $(u_i)_{i \leq n}$  from this uniform distribution, we have the following convergence:

$$(u_i)_{i \leq n} \text{ iid } U, \quad \frac{1}{n} \sum_{i=1}^n f(u_i) \xrightarrow{n \rightarrow \infty} \mathbb{E}_p[f[X]] \quad (24)$$

The convergence is ensured if the random variable  $X = f(U)$  has *bounded variance*. Moreover, the convergence rate is directly related to the variance of  $X$ . If, for example, the infinity norm is bounded:

$$\|f\|_\infty = \sup_{x \in \mathcal{D}} |f(x)| < \infty \quad (25)$$

then the variance of  $f[X]$  is bounded<sup>38</sup>. Then, the approximation error is bounded by:

$$\left| \frac{1}{n} \sum_{i=1}^n f(u_i) - \mathbb{E}_p[f[X]] \right| \leq \frac{\|f\|_\infty}{\sqrt{n}} \quad (26)$$

So, in principle, if we have enough examples/samples, then the average provides a good estimator of the expectation.

### 3.2.1 Volume Calculation

Using the averaging technique, we can compute volumes, for example:

$$|\Omega| = \int_{\mathcal{D}} \mathbf{1}_\Omega(x) dx \quad (27)$$

---

<sup>38</sup> NDJE. We can attempt a small proof. On the one hand, for a random variable  $Y$ ,  $Var[Y] \leq \mathbb{E}[Y^2]$ . Assuming that  $f(\mathbb{E}(X)) = 0$  possibly by redefining  $f$ , then by taking  $Y = f(X)$ , we deduce that  $Var[f(X)] \leq \mathbb{E}[f(X)]$ , which is obviously smaller than or equal to  $\|f\|_\infty$ .

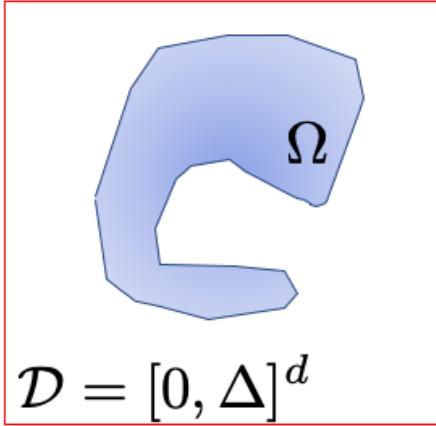


FIGURE 10 – Calculating the volume of  $\Omega \subset \mathcal{D} = [0, \Delta]^d$ .

where  $\mathcal{D} = [0, \Delta]^d$ . By sampling  $(u_i)_i$  uniformly in the box  $\mathcal{D}$ , we obtain

$$|\Omega| \approx \Delta^d \times \frac{1}{n} \sum_{i=1}^n \mathbf{1}_\Omega(u_i) \quad (28)$$

This is known as the hit & miss method (or acceptance-rejection). Note. It is implied that one must be able to determine whether  $u_i$  belongs to the domain  $\Omega$  or not. Note that one can calculate  $\pi/4$  by uniformly sampling points in the square  $[0, 1]^2$  and counting how many fall on average inside the quarter circle of radius 1.

Such calculations are ubiquitous in machine learning, for example, for error computation. One can also consider how AlphaGo or a chess simulator calculates the "value" of a position by considering, from the current position, the proportion of simulated games that are ultimately winning. With such examples, it is clear that simulating all possible games is not feasible, and thus clever strategies are necessary. We will see how such considerations are at work in MCMC methods. Indeed, performing a uniform exploration is not always efficient.<sup>39</sup>

---

39. NDJE. In the notebook [https://github.com/jecampagne/cours\\_mallat\\_cdf/blob/main/cours2023/Monte\\_Carlo\\_Sampling.ipynb](https://github.com/jecampagne/cours_mallat_cdf/blob/main/cours2023/Monte_Carlo_Sampling.ipynb), you will see how importance sampling, for example, helps solve problems of integral computation where there are large "dead" zones if one considers a domain  $\mathcal{D}$  encompassing  $\Omega$  that is poorly adapted.

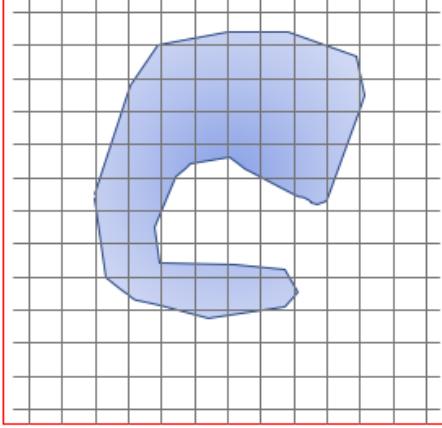


FIGURE 11 – Integral calculation of  $f(x)$  where  $x \in \Omega \subset \mathcal{D} = [0, \Delta]^d$  using a uniform grid.

### 3.2.2 Why Not Perform a Deterministic Calculation?

One can imagine performing an integral calculation using a uniform grid as shown in Figure 11, using the Riemann method. The error incurred is then

$$\varepsilon = \left| \frac{1}{|\mathcal{D}|} \int_{\mathcal{D}} f(x) dx - \frac{1}{n} \sum_{i=1}^n f(x_i) \right| \quad (29)$$

where the  $x_i$  are the points of the grid. We can partition the volume  $\mathcal{D}$  into small volumes, each containing only one point  $x_i$ , which allows us to write:

$$\int_{\mathcal{D}} f(x) dx = \sum_{i=1}^n \int_{\mathcal{V}_i} f(x) dx \quad (30)$$

hence

$$\varepsilon = \left| \sum_{i=1}^n \left( \frac{1}{|\mathcal{D}|} \int_{\mathcal{V}_i} f(x) dx - \frac{1}{n} f(x_i) \right) \right| \quad (31)$$

Noting that  $n|\mathcal{V}_i| = |\mathcal{D}|$ , then

$$\varepsilon = \frac{1}{|\mathcal{D}|} \left| \sum_{i=1}^n \int_{\mathcal{V}_i} (f(x) - f(x_i)) dx \right| \quad (32)$$

The factor  $|\mathcal{D}|$ , equivalent to  $\Delta^d$  seen previously, is not essential for the subsequent discussion. What interests us is whether the asymptotic behavior of deterministic calculation is more or less advantageous compared to a Monte Carlo technique.

Let's assume that  $f \in C^1$  (bounded derivative), then

$$\|f(x) - f(x_i)\| \leq \left( \sup_{x \in \mathcal{V}_i} |f'(x)| \right) \times |x - x_i| \quad (33)$$

Thus

$$\begin{aligned} \varepsilon &\leq \frac{1}{|\mathcal{D}|} \sum_i \int_{\mathcal{V}_i} |f(x) - f(x_i)| \, dx \\ &\leq \frac{1}{|\mathcal{D}|} \sum_{i=1}^d \left( \sup_{x \in \mathcal{V}_i} |f'(x)| \right) \times \left( \sup_{x \in \mathcal{V}_i} |x - x_i| \right) \times |\mathcal{V}_i| \end{aligned} \quad (34)$$

However,  $|\mathcal{V}_i| = |\mathcal{D}|/n$ , so

$$\varepsilon \leq \sup_{x \in \mathcal{D}} |f'(x)| \times \frac{1}{n} \sum_{i=1}^n \sup_{x \in \mathcal{V}_i} |x - x_i| \quad (35)$$

However, if let's say  $|\mathcal{D}| = 1$ , the maximum distance from the points inside each small volume is of the order of  $n^{-1/d}$ . Finally, we obtain the result

$$\varepsilon \leq \sup_{x \in \mathcal{D}} |f'(x)| \times C n^{-1/d} \quad (36)$$

So, in the case of **Monte Carlo calculation, we have a scaling of  $n^{-1/2}$  independent of the dimension**, while **deterministic calculation yields a scaling of  $n^{-1/d}$**  which for large dimension  $d$  becomes exponentially limiting. However, note that this result is obtained by imposing a stronger constraint on the regularity of  $f$ . Therefore, to achieve a given error, one needs a colossal number of samples ( $n \propto \varepsilon^{-d}$ ). **The deterministic viewpoint is confronted with the curse of dimensionality whereas by using the Monte Carlo method, we overcome it.**

Now, why do we have this difference in scaling between the two methods? **The fundamental reason is that in high dimension, the distribution of points in the deterministic**

**grid is not uniform at all.** Clearly, it is not at all suitable for estimating a function for which we know *a priori* only that it is regular. In this case, the uniform distribution is much better suited. But the question remains, can we deterministically design uniform point distributions?

Even by taking a "quincunx" grid, the scaling is  $n^{-2/d}$ , but can we achieve an optimum?

### 3.2.3 Quasi-Monte Carlo Method

The question is to optimally choose the points  $x_i$  such that the error<sup>40</sup>  $\varepsilon$  (Eq. 29) is as small as possible. If Monte Carlo methods use a sequence of random points, quasi-Monte Carlo methods will choose a suitable sequence.

#### Theorem 2 (Quasi-Monte Carlo)

Let the total variation of the function be defined as

$$\|f\|_{TV} = \int_{\mathbb{R}^d} \|\nabla f(x)\| dx \quad (37)$$

which we assume to be finite. Under this assumption, we can define  $\{x_i\}_{i \leq n}$  such that

$$\varepsilon \leq \|f\|_{TV} \times C_{n,d} n^{-1} \quad (38)$$

with

$$C'_d (\log n)^{d-1} \leq C_{n,d} \leq C_d (\log n)^{d-1/2} \quad (39)$$

So, at best we have a bound of  $n^{-1}(\log n)^{d-1/2}$  which is **better than that obtained with the Monte Carlo technique but for this  $n \geq e^d$  due to the dependence of the constants "C" on the dimension**, which makes these methods much less efficient than the Monte Carlo method.

Certainly, the quasi-Monte Carlo method appears to be efficient due to its scaling in  $n$ , but the constants are huge and we also assume a fairly strong regularity assumption on  $f$ , as it underlies a form of sparsity. In practice, "mixed" methods are used in this context

---

40. NDJE. In the course,  $|\mathcal{D}|$  is implicitly taken as 1 in the formulas.

where we start with a deterministic grid that is randomly refined to improve the scaling of the constants. But it remains true that the uniformity of the distribution of grid points is the most difficult aspect. This is why calculations via Monte Carlo are much easier to implement. However, what are their limits?

### 3.2.4 Limits of the Monte Carlo Method

In high dimensions, as mentioned earlier, we have **concentration phenomena**<sup>41</sup>. Therefore, we will want to calculate integrals for which the measure of the support  $\Omega$  of the functions is almost zero. If the integral is non-zero (let's say  $f$  is positive), then this implies that  $f(x)$  has large amplitudes such that the integral of  $f^2(x)$  is very large (variance), as is  $\|f\|_\infty$ . Therefore, the error bound by Monte Carlo  $\varepsilon$  (Eq. 26) is very large, requiring many samples.

Another way to see the problem generated by data concentration is to consider the rejection method. If we enclose the domain where the data concentrate with a "box"  $\mathcal{D}$ , the efficiency of the method is governed by the ratio  $|\Omega|/|\mathcal{D}|$  which is very small. Therefore, we have a practical problem of generating many points to achieve a given error.

To solve this problem,  $\mathcal{D}$  must adapt<sup>42</sup> as closely as possible to  $\Omega$  (Fig. 12). For this, it is necessary to **find probability distributions that adapt to the geometry**, and they must be samplable. In a sense, we can then perform the following calculation

$$\int_{\mathbb{R}^d} f(x) dx = \int_{\mathcal{D}} \underbrace{\frac{f(x)}{p(x)}}_{w(x)} p(x) dx \approx \frac{1}{n} \sum_{i=1}^n w(x_i) \quad x_i \sim p(x) \quad (40)$$

So, the problem is to find probability distributions  $p(x)$  that can approximate an arbitrary domain  $\Omega$ , and from which samples can be drawn.

---

41. NDJE. See Courses 2022, 2023, on the theme of Shannon's typical sets.

42. NDJE. Hence the Importance Sampling method, an example of which is given in the notebook of the footnote on page 39.

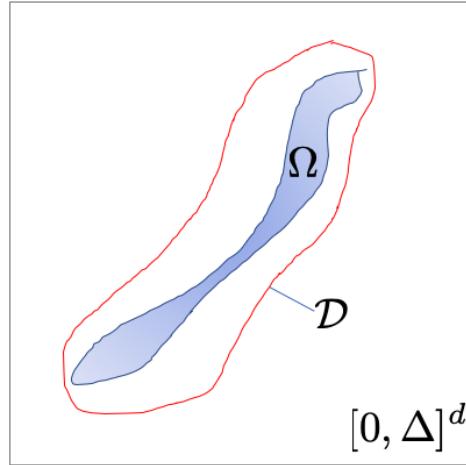


FIGURE 12 – Schematic representation of adjusting to the geometry of the data space  $\Omega$ , of the enclosing space in which points are sampled: transition from  $[0, \Delta]^d$  to  $\mathbb{D}$ .

### 3.3 Probability Modeling, Learning/Inference

We want to approximate a probability  $p(x_1, \dots, x_d)$  as well as  $p(y|x_1, \dots, x_d)$  the conditional probability of  $y$  given the data  $x$ , in high dimensions. The previous section suggests that *a priori* this is a difficult problem.

#### 3.3.1 Small Example: Spam Detection

Consider the problem of spam detection: given an email composed of words from the French language, we want to determine if it is spam ( $y = 1$ ) or not ( $y = 0$ ). We encode  $x_i = 1$  if word  $i$  (from the dictionary) is present in the email (and  $x_i = 0$  otherwise). So, we have a problem with  $d$  binary variables, and the combinatorics tell us that there are  $2^d$  possibilities. It is not feasible to compute all corresponding probabilities when  $d$  represents the size of the current French language dictionary. Therefore, approximations need to be made.

In the case of the spam problem, we use the "**naive Bayes assumption**". According

to Bayes' formula, we have

$$p(y|x_1, \dots, x_d) = \frac{p(x_1, \dots, x_d|y)p(y)}{p(x_1, \dots, x_d)} \quad (41)$$

and the assumption is to write

$$p(x_1, \dots, x_d|y) = \prod_{i=1}^d p(x_i|y) \quad (42)$$

meaning that the words are independent of each other under the assumption of being part of spam or not. To obtain the score of the email, we maximize the *a posteriori* probability, so we want to obtain

$$y^* = \operatorname{argmax}_y p(y) \prod_{i=1}^d p(x_i|y) \quad (43)$$

In this approximation framework to estimate  $p(x_i|y)$ , there are 4 states to consider for the pair  $(x_i, y)$  but actually only 2 degrees of freedom (note:  $p(x|y = 1) + p(x|y = 0) = 1$ ). Ultimately, **we have  $d$  probabilities to estimate which makes a significant difference**.

In practice, this independence-based method is quite effective for the spam detection problem. However, when it comes to **image classification**, the assumption of considering only  $p(x_i|y)$  raises the question: given the value of pixel  $i$  (i.e.,  $x_i$ ), are we dealing with an image of a dog vs. cat, car vs. airplane, etc.? It is clear that **the assumption does not hold**. Instead, we need to be able to **analyze the correlations** of pixels to identify structures.

### 3.3.2 Learning: Finding the Best Approximation

To address problems like image generation/classification, as discussed during the introductory session, we need to use much more sophisticated **parametrized families**  $\{p_\theta(x)\}$ , such as **Markov random fields**. We will explore this further.

Once a parametrized family of probabilities is defined, we need to find the best  $\theta$  such that  $p_{\theta^*}(x)$  best approximates  $p(x)$ . There are many possible approaches that involve

minimizing an objective function, such as the Kullback-Leibler divergence<sup>43</sup> defined as

$$D_{KL}(p||p_\theta) = \int p(x) \log \frac{p(x)}{p_\theta(x)} dx \geq 0 \quad (44)$$

and  $D_{KL} = 0$  if and only if  $p = p_\theta$ . We can rewrite the expression of  $D_{KL}(p||p_\theta)$  as follows

$$D_{KL}(p||p_\theta) = \underbrace{\int p(x) \log p(x) dx}_{-\mathbb{H}[p]} - \underbrace{\int p(x) \log p_\theta(x) dx}_{\mathbb{E}_p[\log p_\theta(x)]} \geq 0 \quad (45)$$

where  $\mathbb{E}_p[\log p_\theta(x)]$  is the likelihood and  $\mathbb{H}[p]$  is the differential entropy of  $p$  in the sense of Shannon<sup>44</sup>, which is a constant with respect to the problem of optimizing the value of  $\theta$ . The idea dates back to R. Fisher because **maximizing the likelihood, i.e., the ability of the distribution  $p_\theta(x)$  to account for the data distribution, minimizes the Kullback-Leibler divergence**. If we have data  $(x_i)_{i \leq n}$  supposed to be representative of the underlying probability  $p(x)$  then

$$\mathbb{E}_p[\log p_\theta(x)] \approx \frac{1}{n} \sum_{i=1}^n \log p_\theta(x_i) \quad (46)$$

The objective is therefore to maximize this quantity (or minimize it if we consider the "-" sign).

**The Bayesian hypothesis adds an assumption about the possible values of  $\theta$  (prior)** which is a form of regularization in terms of estimation. In fact, we would like to maximize the *posterior probability*  $p(\theta|x)$  (the best  $\theta$  given the data  $x$ ). Recalling Bayes' formula (Eq. 4):

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \quad (47)$$

optimizing  $p(\theta|x)$  amounts to optimizing the product of the *likelihood* ( $p(x|\theta)$ ) and the *prior* ( $p(\theta)$ ). Now, the likelihood is by construction what we denoted  $p_\theta(x)$ . Maximizing  $p(\theta|x)$  therefore amounts to maximizing the log, and since we have a sample basis, we can maximize on average. It follows that

$$\theta^* = \operatorname{argmax}_\theta \mathbb{E}_p[\log p(\theta|x)] = \operatorname{argmax}_\theta (\mathbb{E}_p[\log p_\theta(x)] + \log p(\theta)) \quad (48)$$

---

43. NDJE. see the 2019 Course Sec. 7.2.3 for example

44. NDJE. see, for example, Course 2022 Sec. 5 and Course 2023 Sec. 5.1

We therefore add *a priori* information that we have about the values of  $\theta$ . This information acts in Bayesian optimization like regularization<sup>45</sup> (in L2 norm *ridge regression*<sup>46</sup>, or in L1 norm to increase sparsity in the parameters) in a deterministic algorithm of linear regression (or SVM)) or in parameter optimization of a neural network, to **avoid overfitting**.

Once the objective function to be optimized is defined, the optimization method is usually a **gradient descent** (note: minimization taking into account the "-" sign) that we have already covered in previous years' courses<sup>47</sup>.

### 3.3.3 Inference

Consider the spam detection problem, inference consists, for example, of calculating the probability that an email containing the words  $k$  and  $j$  is spam or not. Thus,

$$p(y|x_k, x_j) = \frac{p(y, x_k, x_j)}{p(x_k, x_j)} = \frac{\int p(y, x_1, \dots, x_d) \prod_{i \neq k,j} dx_i}{\int p(x_1, \dots, x_d) \prod_{i \neq k,j} dx_i} \quad (49)$$

The two integrals are in high dimension and Monte Carlo techniques are therefore used. As S. Mallat tells us, in this field of mathematics, "we spend our lives calculating integrals".

### 3.3.4 Which type of modeling to choose? Examples

This is the most open, still within the realm of research, and the most difficult part: which families of probabilities are best suited to approximate data distributions. It is in this problem that **neural networks have made the difference**.

In the 2000s, S. Mallat tells us, roughly for applications, modeling was done using **Gaussian models**. In this type of modeling, we use the following expression<sup>48</sup>, assuming that  $\mathbb{E}[x] = 0$  for simplicity of notation:

$$p(x) = Z^{-1} \exp \left( -\frac{1}{2} x^T K x \right) \quad (50)$$

---

45. NDJE. See Course 2018 Sec. 7.3.2 and/or Course 2019 Sec. 7.2.4.2

46. NDJE. e.g., if we seek to minimize  $\theta^T \phi(x) + \lambda \|\theta\|^2$  this is equivalent to defining a prior of the form  $p(\theta) \propto e^{-\lambda \|\theta\|^2}$ .

47. NDJE. see, for example, Course 2018 Sec. 10.1

48. NDJE. See for example Course 2022 Sec. 2.6 and/or Course 2023 Sec.8.1

where  $K$  is a kernel (symmetric positive definite  $d \times d$  matrix), and thus, in terms of energy, we have a quadratic function in  $x$ . It can be shown<sup>49</sup> that  $K$  is related to the covariance matrix  $C = \mathbb{E}_p(xx^T)$ , i.e.,<sup>50</sup>  $C = K^{-1}$  (centered second-order moments), and its elements are the parameters  $\theta$  to be optimized.

A particularly important example used, for instance, in signal processing is the **stationarity assumption** (translation invariance). If we index by  $i$  the coordinate of a pixel in an image or a sample in a time frame, according to this assumption:

$$\forall k \geq 0, \forall \tau, \quad p(x_{i_1-\tau}, x_{i_2-\tau}, \dots, x_{i_k-\tau}) = p(x_{i_1}, x_{i_2}, \dots, x_{i_k}) \quad (51)$$

This assumption is often verified. S. Mallat, for example, illustrates it by considering the case of an image without a reference point; we can also mention in cosmology that the properties of the cosmic microwave background (CMB) are approximately invariant in the direction pointed by the instruments (translation in the right ascension-declination space). However, centered face images like passport photos are not stationary at all.

In this stationary framework, we have the following property:

$$\mathbb{E}_p(x_i x_j^T) = \mathbb{E}_p(x_{i-j} x_0^T) = f(i - j) \quad (52)$$

meaning that **the covariance depends only on the translation parameter**, thus it is a Toeplitz matrix (diagonal elements identical).

Now, since  $K = C^{-1}$  is symmetric positive definite, we can look at the distributions in the base that diagonalizes it. This is the **principal component analysis (PCA) base**. **The stationary assumption indicates that the PCA base is the Fourier base**<sup>51</sup>. Indeed, neglecting edge effects, if  $e$  is an eigenvector of  $C$ , then by definition:

$$Cx = \lambda x \quad \Rightarrow \forall j, \quad \sum_i C(i - j)e_i = \lambda e_j = \sum_k C(k)e_{k+j} \quad (53)$$

hence the  $k$ -th component of a vector in the base is a sine wave  $e_k(\omega) = e^{ik\omega}$ , and the positive eigenvalue  $\lambda$  can be written as a variance  $\sigma_\omega^2$ , which is nothing but  $\hat{C}(\omega)$ , the

---

49. NDJE. here  $x = (x_1, \dots, x_d)^T$  so  $x$  is of dimension  $d \times 1$ .

50. NDJE *hint*:  $K$  is diagonalizable with  $P$  an orthogonal matrix, such that  $K = P^T D P$ , and we perform a change of variable  $y = Px$ .

51. NDJE. See Course 2021 Sec. 4.4

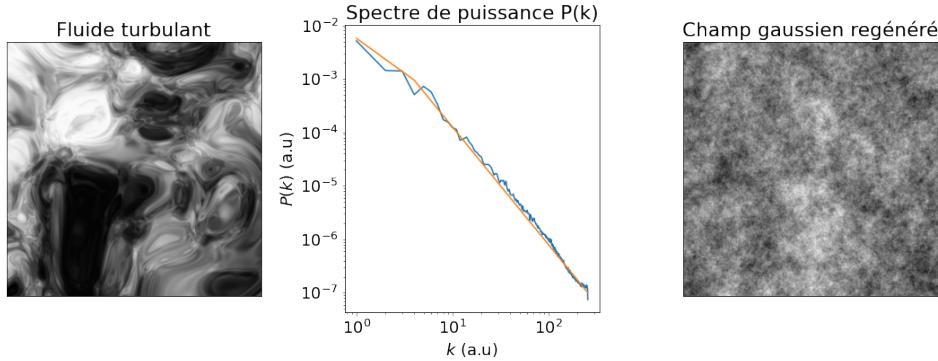


FIGURE 13 – On the left, an image of turbulent fluid in a pipe (Credit: Piotr Siedlecki/public domain); in the center: the power spectrum derived from the image ( $\propto k^{-2.2}$ ) where  $k$  here is the norm of the wave vector which is the notation commonly used in Cosmology; on the right: a Gaussian field generated from this power spectrum.

power spectrum (or spectral power)<sup>52</sup>. This is the basic model that essentially states that **probability distributions are concentrated - typical sets are ellipsoids - along the principal axes of the covariance matrix**.

However, **this model is not sufficient**, and to illustrate this, let's take the example of turbulence<sup>53</sup>. In Figure 13, on the left, we have an example image of turbulent fluid<sup>54</sup>, in the center its power spectrum, and on the right, a realization of a Gaussian field generated from this power spectrum. To do this, we simply need to measure the two-point correlation function (Fourier transform of the power spectrum), that is, estimate the covariance matrix. If we see granularity, **we have lost all the structures** of the original image. Yet note that vortices can occur anywhere in the image, so **we are indeed dealing with a stationary field**. Therefore, something is missing in the modeling. But what?

In fact, **the Gaussian model corresponds to a maximum entropy model**<sup>55</sup>, that is, a form of maximum disorder given the information it has on the values of the cova-

52. NDJE. In cosmology, the power spectrum is rather denoted  $P(k)$  where  $k$  is the wave vector norm.

53. NDJE. This is an example from the 2022 Course Sec. 4.5, which I have made available in the notebook [https://github.com/jecampagne/cours\\_mallat\\_cdf/blob/main/2023/gaussian\\_vs\\_turbulent\\_fow.ipynb](https://github.com/jecampagne/cours_mallat_cdf/blob/main/2023/gaussian_vs_turbulent_fow.ipynb).

54. NDJE. Image from the article <https://phys.org/news/2015-10-key-features-transition-liquid-smooth.html>.

55. NDJE. This is Gibbs's theorem seen in the 2023 Course Sec. 7.6

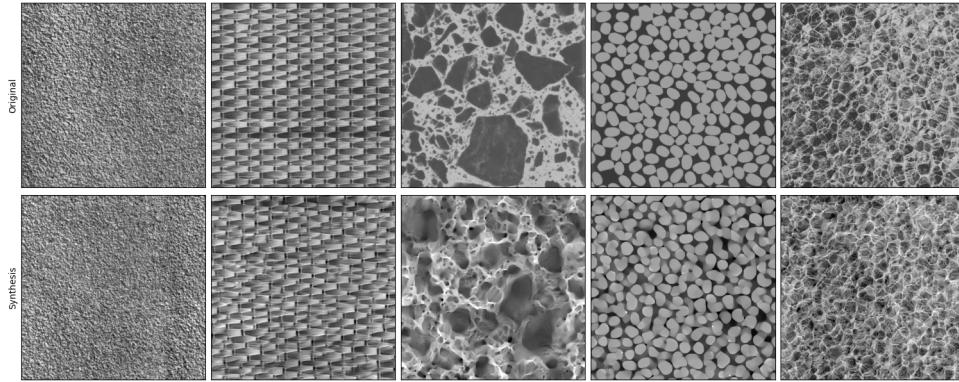


FIGURE 14 – Example of textures that can be synthesized by going beyond Gaussian field modeling (wavelet networks).

riance matrix coefficients. However, turbulence is structured and thus **typical sets are not ellipsoids**.

By going further into modeling, during the introduction S. Mallat showed us that we could **synthesize stationary fields**, such as physical fields (Fig. 6), or textures like those in Figure 14 using wavelet networks<sup>56</sup>.

The current research domain is to understand how to **synthesize non-stationary (non-ergodic) fields**. S. Mallat invites us to look at the seminars by Marylou Gabré who talks about models in the framework of Statistical Physics and Markov fields, and by Francis Bach who discusses face and other image modelings with Score Diffusion models. **The two big differences lie in the estimation technique, but above all in the Markov vs. Neural Network modeling.**

S. Mallat shows us an application of the latest modeling, namely **inpainting** (Fig. 15) which presents itself here as the generation of an image  $y$  according to the probability  $p(y|x)$ , where  $x$  is a degraded image and  $p$  is a model trained with a very large database of images. The image  $y$  is a reconstruction of the image  $x$  (note: it is not necessarily the original image, but the chosen examples illustrate the possibilities). The model has captured all forms of regularity.

---

<sup>56</sup>. NDJE. You can see how this is done via the notebook [https://github.com/jecampagne/cours\\_mallat\\_cdf/blob/main/2023/TextureSynthesis.ipynb](https://github.com/jecampagne/cours_mallat_cdf/blob/main/2023/TextureSynthesis.ipynb).



FIGURE 15 – Illustration of *inpainting*, where from a model trained with a very large database of images, it manages to generate an image  $y$  from an image with whitened parts.

We can also perform **denoising** of an image as shown in Figure 16. This is a cell, and typically in X-ray imaging, we do not want to expose tissues to high doses, so the signal-to-noise ratio is not optimal.

Despite these nice results, there is a big caveat or even a "danger" as S. Mallat puts it. Usages are very cautious, especially in medical imaging, because of **hallucination phenomena**. Indeed, with diffusion models, similar to denoising, starting from noise, one can gradually make an image appear that is geometrically on a typical space (but not an ellipsoid) as shown in Figure 17. The question arises: what can be said about the generated image? In the medical case, one could reveal tumors, for example, or not. The problem is to avoid these hallucination phenomena.

S. Mallat then shows us (Fig. 18) an example of **component separation** in audio that he presented during the 2019 course, dating back to 2018<sup>57</sup>. This is the "Cocktail Party challenge" where one tries to separate speeches spoken simultaneously by two speakers with the same voice timbre. The contribution of neural networks in this field has brought about a real revolution, as he tells us, because roughly for 30 years, people had been

---

<sup>57</sup>. NDJE. Refer to the 2019 Course Sec. 2.2.2, and for the audio, I invite you to watch the video of the 2024 course around 1:29:40.

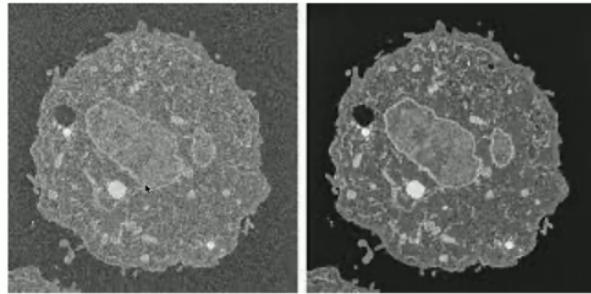


FIGURE 16 – Illustration of denoising. We observe  $x$  which is a superposition of  $y$  and noise, and we want to extract  $y$ , or rather a sample of  $p(y|x)$ .

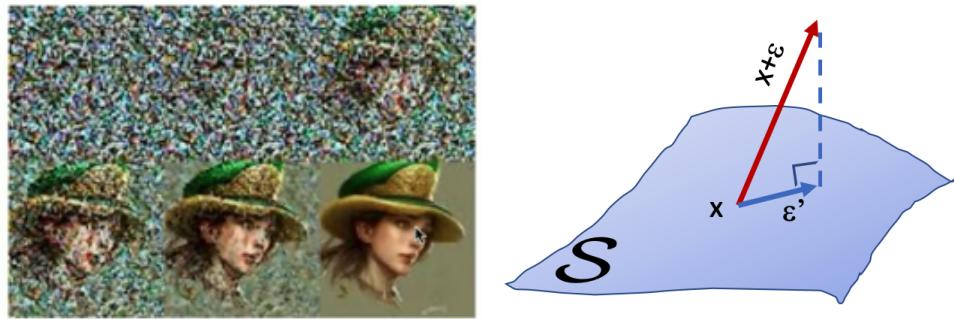


FIGURE 17 – Illustration of generation by a diffusion model resembling denoising but not quite. Once trained by core matching, the model generates an image from noise. It somehow projects a completely noisy image as if it were decomposable into a "noise-free" image  $x$  belonging to a typical set ( $S$ ) and noise. However, what can be said about the  $x$  chosen by the model?

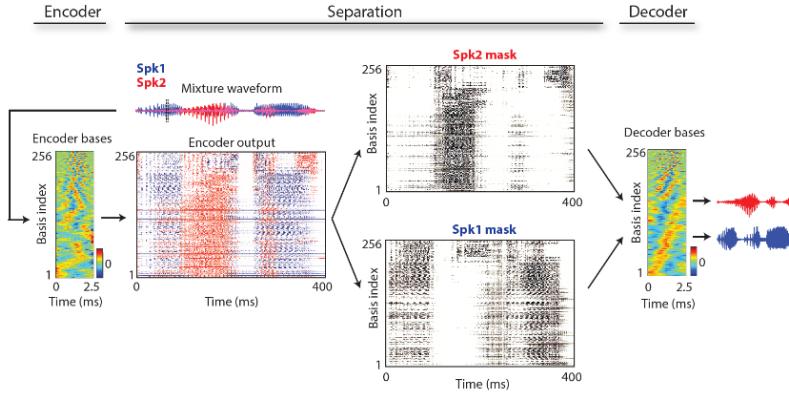


FIGURE 18 – A result from 2018 (See 2019 Course) showing how one can solve the separation of speeches from two speakers with the same voice type.

trying to solve the problem in signal processing from spectrograms and it did not yield promising results.

To conclude and introduce Michele Sebag's seminar, S. Mallat indicates that current neural network models do not implement directional graphs that would support **causal modeling**. In practice, we currently do not know how to learn such structures. Therefore, in the context of the course, we will use non-causal models.

## 4. Lecture of January 31st

During this session, we will address the *modeling* and *approximation*<sup>58</sup> through the fundamental notion of **Markov Random Fields**. This is the tool that allows us to understand, with relatively low-dimensional models, the interactions between variables. In this context, approximation and optimization are two facets of learning. Recall that we must choose a parameterized model that potentially approximates the true probability distribution, and we must be able to determine the best set of parameters.

We find the connection with Statistical Physics seen in the 2023 course. The principle

---

58. NDJE. Ultimately, S. Mallat did not have time to address this topic, which will be the subject of the next session.

is to model the **interactions between variables** while explaining the notion of **conditional independence**.

## 4.1 Causal and Non-causal Schemes

Let's take a simple example of the voting behavior of four people (A, B, C, D) who will interact and therefore mutually influence each other. To model the influence between 2 people (X, Y) at the time of voting, we can say that if the votes of X and Y are identical then  $\Phi(X, Y)$ , which represents the interaction between X and Y, is large, and conversely if the votes are different then  $\Phi(X, Y)$  is small. In the end, what we are looking for is the joint probability  $p(A, B, C, D)$ , modeled for example as:

$$p(A, B, C, D) = \frac{1}{Z} \Phi(A, B) \Phi(B, C) \Phi(C, D)$$

with  $Z$  as a normalization constant. Before the vote, influences will play their roles, and we can say that there is a form of equilibrium that will occur (assuming it happens before the vote). The most probable vote reflects the interactions between the different members. However, to be able to calculate  $p(A, B, C, D)$ , we must be able to determine the constant  $Z$ . Moreover, if we want to understand the vote of A, we need to understand the exchange of information between the different members (phenomenon of propagation). This is more complicated than a directional model.

A *directional model* (Fig. 19) is at work, for example, if we know that A influences B and C, which in turn influences D. In this case, we have (*chain rule*):

$$p(A, B, C, D) = p(A|B, C, D)p(B, C|D)p(D)$$

Then, sampling a configuration (A,B,C,D) is done by finding a configuration of D alone, then a configuration of the pair (B,C) given that of D, and finally a configuration of A given that of (B,C,D). In a way, by reversing the flow of influences, we can sample the joint probability.

The first model is typically a **non-directional Markov Field**, which is more complicated to sample than the second, representing a **directional Markov Field**. However, establishing directional relationships is much more complicated: it is easier to know if

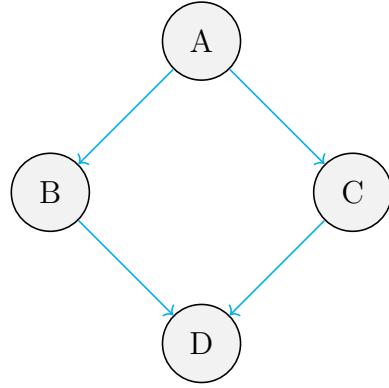


FIGURE 19 – Directional Model.

two variables are correlated than to know the causal pattern between them (see Michèle Sebag's conference). **All current Machine Learning is based on the non-directional framework**, including neural networks and even very large language systems.

## 4.2 Markov Field

### 4.2.1 Definition and Properties

The definition of a Markov Field (*Markov Random Field*) is as follows:

**Definition 1 (Markov Field/MRF)**

A Markov Field is a joint probability distribution defined on a non-directional graph  $G$ , whose nodes  $(x_k)_{k \leq d}$  (random variables) follow a probability density that can be written as:

$$p(x_1, x_2, \dots, x_d) = Z^{-1} \prod_{c \in \mathcal{C}} \Phi_c(x_c) \quad (54)$$

where  $x_c = (x_{i_1}, \dots, x_{i_k})$  and  $(i_1, \dots, i_k) \in c$ , and "c" is called a "clique". The constant  $Z$  ensures normalization:

$$\int p(x_1, x_2, \dots, x_d) \prod_{i=1}^d dx_i = 1 \quad (55)$$

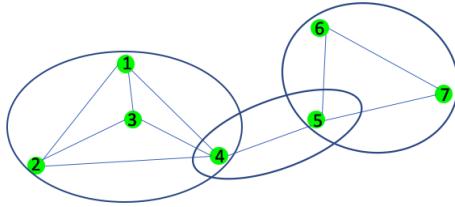


FIGURE 20 – Example of an undirected graph comprising 7 "1-vertex" cliques (vertices), 10 "2-vertex" cliques (edges), 5 "3-vertex" cliques, and 1 "4-vertex" clique. The three surrounded cliques form the set  $\mathcal{C}$ .

*The probability density  $p$  is a Gibbs distribution.*

The term *clique* in graph theory refers to a fully connected subgraph (Fig. 20). The definition of  $\mathcal{C}$  allows for **factorization**. This factorization is important because even if two distant variables in the graph are dependent (i.e., there is a path connecting them), we model the interactions within the cliques of  $\mathcal{C}$ , which are themselves **local**. Thus, we seek a **low-dimensional representation** centered on the local nature of interactions (as done in multi-scale modeling).

Within the framework of Markov fields, we have a fundamental property. Before stating it, let's have a brief reminder<sup>59</sup>:

- Let  $x_1$  and  $x_2$  be two random variables, they are **probability independent** iff

$$p(x_1, x_2) = p(x_1)p(x_2)$$

- Let  $x_1$  and  $x_2$  be two random variables, and  $y$  be a third random variable that can influence the first two, the **conditional independence** of variables  $x_1$  and  $x_2$  then can be expressed as

$$p(x_1, x_2|y) = p(x_1|y)p(x_2|y)$$

For example, in this context, we deduce:

$$p(x_1|x_2, y) = \frac{p(x_1, x_2, y)}{p(x_2, y)} = \frac{p(x_1, x_2|y)p(y)}{p(x_2|y)p(y)} \xrightarrow{\text{cond. indep.}} p(x_1|y)$$

---

59. NDJE. See course 2022/2023

meaning that  $x_2$  does not provide any information about  $x_1$  if we already know  $y$ . This conditional independence is denoted by  $x_1 \perp x_2 | y$ .

The Markov property then gives us:

**Property 1 (Markov)**

For a Markov field, if there is no clique that includes two variables  $(x_i, x_j)$  (i.e., there is no edge directly connecting the two variables), then variable  $x_i$  has a conditional independence with respect to  $x_j$  such that

$$x_i \perp x_j | \{x_k\}_{k \neq i,j}$$

meaning

$$p(x_i, x_j | \{x_k\}_{k \neq i,j}) = p(x_i | \{x_k\}_{k \neq i,j}) \times p(x_j | \{x_k\}_{k \neq i,j}) \quad (56)$$

For example, in Figure 20, variables  $x_1$  and  $x_6$  are in the situation described by the property, and therefore if we know all the other variables (conditioning),  $(x_1, x_6)$  are independent. In other words,  $x_1$  and  $x_6$  do not directly influence each other, and the influence is through the other variables of the graph.

**Proof 1.** We are dealing with a graph supporting a Markov field. The set  $\mathcal{C}$  can be expressed as a disjoint union of subsets of cliques as follows:

$$\mathcal{C} = \mathcal{C}_{i,j} \cup \mathcal{C}_{i,j'} \cup \mathcal{C}_{i,j} \cup \mathcal{C}_{i,j'} \quad (57)$$

where the subsets are indexed based on whether  $x_i$  and/or  $x_j$  belong to the set or not. The assumption then states that  $\mathcal{C}_{i,j}$  is the empty set. Thus, we can write

$$\begin{aligned} p(x_i, x_j | \{x_k\}_{k \neq i,j}) &= \frac{p(\{x_k\}_{k \leq d})}{p(\{x_k\}_{k \neq i,j})} \\ &= \frac{\prod_{c \in \mathcal{C}_{i,j}} \Phi_c(x_c) \prod_{c \in \mathcal{C}_{i,j'}} \Phi_c(x_c) \prod_{c \in \mathcal{C}_{i,j}} \Phi_c(x_c)}{p(\{x_k\}_{k \neq i,j}) \prod_{c \in \mathcal{C}_{i,j}} \Phi_c(x_c) \prod_{c \in \mathcal{C}_{i,j'}} \Phi_c(x_c) \prod_{c \in \mathcal{C}_{i,j}} \Phi_c(x_c) \times dx_i dx_j \prod_{k \neq i,j} dx_k} \end{aligned} \quad (58)$$

We can distribute the different elements of the integral, which factorize, thus revealing:

$$p(\{x_k\}_{k \neq i,j}) \int \prod_{c \in \mathcal{C}_{i,j}} \Phi_c(x_c) \prod_{k \neq i,j} dx_k = \prod_{c \in \mathcal{C}_{i,j}} \Phi_c(x_c) \quad (59)$$

which simplifies with the corresponding term in the numerator. Moreover, the integrals depending on  $x_i$  for one and on  $x_j$  for the other also factorize. Therefore,

$$\begin{aligned} p(x_i, x_j | \{x_k\}_{k \neq i,j}) &= \frac{\prod_{c \in \mathcal{C}_{i,j}} \Phi_c(x_c)}{\int \prod_{c \in \mathcal{C}_{i,j}} \Phi_c(x_c) dx_i} \times \frac{\prod_{c \in \mathcal{C}_{i,j}} \Phi_c(x_c)}{\int \prod_{c \in \mathcal{C}_{i,j}} \Phi_c(x_c) dx_j} \\ &= p(x_i | \{x_k\}_{k \neq i,j}) \times p(x_j | \{x_k\}_{k \neq i,j}) \end{aligned} \quad (60)$$

The last equality is obtained by reinjecting into each term of the first line, in the numerator, the right member of Equation 59, and in the denominator, the left member of the same equation. Therefore, we have obtained the equation indicating the conditional independence of  $(x_i, x_j)$  given the values of  $\{x_k\}_{k \neq i,j}$ . ■

## 4.2.2 Two Generic Examples

### 4.2.2.1 Gaussian Process

Let's see some examples of applications, starting with **the Gaussian process**. Note by the way that for a Markovian field we can write

$$\log p(x) = -\log Z + \log \prod_{c \in \mathcal{C}} \Phi_c(x_c) \quad (61)$$

Now for a Gibbs distribution traditionally we have

$$\log p(x) = -\log Z - U(x) \quad (62)$$

We thus identify **the term of the cliques as the energy term**  $-U(x)$ . In the Gaussian case  $U(x) = \frac{1}{2}x^T K x$ . We know that the matrix  $K$  is related to the covariance matrix  $K^{-1} = \mathbb{E}(xx^T)$ . The Markov property tells us that if  $K_{i,j} = 0$  then  $x_i \perp x_j | \{x_k\}_{k \neq i,j}$ . So, we can directly view  $K$  as the interaction matrix.

Another example used especially in Physics, concerns the case where  $K = \Delta$  (**Laplacian**). Let's say  $x$  are pixels of an image (underlying grid) then the discrete Laplacian can be represented as

$$\Delta = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (63)$$

which represents the interactions of a pixel with its neighbors. **Here only its closest neighbors count.** So, maximizing the probability implies minimizing the energy which amounts to maximizing the Laplacian interaction favoring regularities, but the values of  $x$  will still fluctuate. Now, we can rewrite the interaction term as<sup>60</sup> (subject to boundary conditions) as

$$-\frac{1}{2}x(u)^T \Delta_u x(u) = -\frac{1}{2} \int x(u) \nabla_u x(u) du = \frac{1}{2} \int |\partial_u x(u)|^2 du = \frac{1}{2} \sum_{i,j \in \mathbb{C}} |x_i - x_j|^2 \quad (64)$$

where index pairs  $i, j$  are such that  $j$  are taken in cliques corresponding to the closest neighbors of  $i$  (we can note this property as  $(i, j)$ ). The probability becomes

$$p(x) = Z^{-1} \exp \left\{ -\frac{1}{2} \sum_{(i,j)} |x_i - x_j|^2 \right\} \quad (65)$$

which has **the expression of a Brownian motion**<sup>61</sup>.

#### 4.2.2.2 $\phi^4$ Theory

As a second example, S. Mallat shows us a modeling of what is called **the  $\phi^4$  theory** in field theory which is derived from **the Ising model**<sup>62</sup>. The  $\phi^4$  theory stems from the

---

60. NDJE. In Statistical Physics (field theory), in fact, we do the opposite because the term  $\partial_\mu \phi(x) \partial^\mu \phi(x) = |\partial_\mu \phi(x)|^2$  is nothing but the kinetic energy term.

61. NDJE. In its simple form, a Brownian motion can be simulated by a process such as  $X_t = X_{t-1} + Z_t$  where  $Z_t$  is a *v.a* following a normal distribution, so  $p(X_t, X_{t-1}) \propto \exp\left\{-\frac{1}{2}(X_t - X_{t-1})^2\right\}$ .

62. NDJE. The problem that Lars Onsager (1903-76) solved exactly in 1944 is the since famous 2D Ising model: this interacting spin model was introduced by Wilhelm Lenz (1888-1957) in 1920 and his student Ernest Ising (1900-98) had solved it only in 1D and had not been able to find a phase transition. Onsager's exact solution allowed to understand its meaning and the study of critical exponents and the development in Statistical Mechanics of the Renormalization Group Equation Theory which will lead to many results as in the Standard Model Theory of Elementary Particles (Brout-Englert-Higgs phenomenon

works of Vitaly Lazarevich Ginzburg (1916-2009) and Lev Davidovich Landau (1908-68). S. Mallat discusses a model slightly different from the one classically studied. On a  $L \times L$  grid, the energy is given by

$$U(x) = \frac{\beta}{2} \sum_{(i,j)} (x(i) - x(j))^2 + \sum_i (x(i)^2 - 1)^2, \quad p(x) \propto e^{-U(x)} \quad (66)$$

In this framework, the potential is fixed, but what varies is the weighting factor  $\beta$  of the Brownian term compared to the potential term. Depending on the value of this parameter (the inverse of some sort of temperature), and in the limit as  $L \rightarrow \infty$ , there is a competition between the two terms. A **phase transition** occurs when  $\beta = \beta_c \approx 0.68$ . Indeed,

- When  $\beta \ll \beta_c$ , the kinetic term (Laplacian) becomes negligible (it disappears from the modeling when  $\beta = 0$ ) and only the potential terms matter, which are separable. Then there is independence between the different variables  $x_i$ :

$$p(x) \propto \prod_i \exp\{-V(x(i))\} \quad (67)$$

the system is disordered.

- When  $\beta = \beta_c$ , the Brownian term tends to favor smooth configurations while the potential term tends to make the  $x_i$  take their values in  $\{-1, +1\}$ . It is a system where order propagates on scales much larger than the local scale (this is also reflected in the form of the power spectrum for  $\beta = \beta_c$ ). This is a competition between order/disorder that gives rise to a **phase transition** phenomenon.
- Finally, for  $\beta \gg \beta_c$ , the configurations are only guided by short-range interactions. If the initial configuration has more than 1, this value tends to spread to all  $x_i$  (and symmetrically if initially there are more  $-1$ ). There are small local fluctuations around these two values.

Simulations of the different situations depending on the value of  $\beta$  with respect to  $\beta_c$  are shown in Figure 21. **While the model is purely local, correlations beyond nearest neighbors are observed in the simulations where  $\beta \geq \beta_c$ .** We then witness the extension of areas where the values of  $x_i$  take the values  $\pm 1$ .

---

Nobelized in 2013). The  $\phi^4$  theory is a generalization for spins with continuous values.

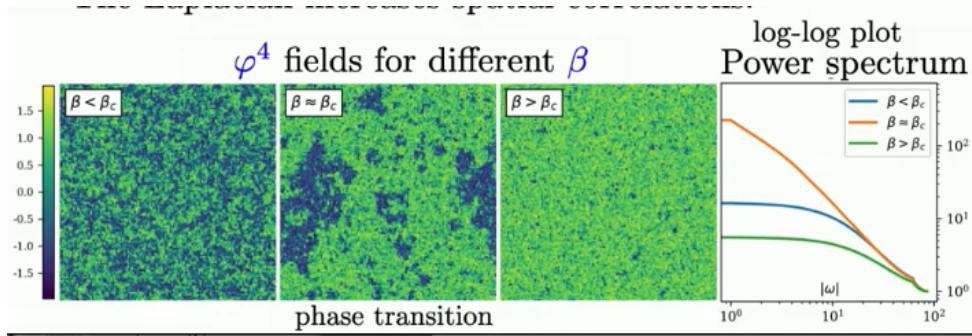


FIGURE 21 – Simulations of configurations of the model Eq. 66.

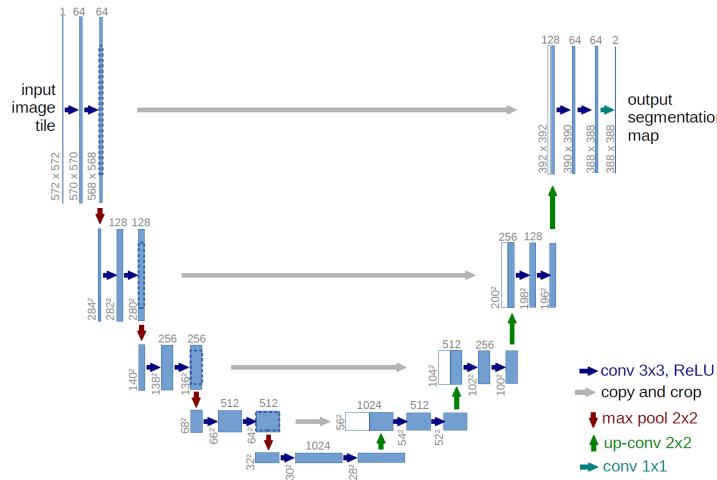


FIGURE 22 – Multi-scale architecture of a U-Net.

The important point is that **to understand the emergence of structures which are the manifestation of interactions between fields at long range, it is necessary to separate the different scales**, as there seems to be a hierarchical organization of interactions, a concept that emerged in Physics with the notion of **Renormalization Group**<sup>63</sup>. What is remarkable is that these ideas developed in Physics are found in neural networks. For example, a U-Net takes an image, decomposes it into different scales, and its architecture allows for modeling **multi-scale interactions** (Fig. 22). Ultimately, if large generative models that manipulate high-dimensional probability distributions are learnable, it is because **there are underlying factorizations at work**. S. Mallat gives us an example with face generation, where one can impose local interactions between pixels due to multi-scale phenomena similar to those in Physics.

### 4.3 Independence Properties

We will study the notion of independence by taking groups of variables as illustrated in Fig. 23. Typically, when modeling an image, instead of considering interactions between pixels, we are interested in interactions between blocks consisting of structures. Note in the example that not all variables  $x_i$  in block  $X$  are directly connected (in interaction) with variables  $x_j$  in block  $Z$ .

#### Property 2

*In a Markov field, if there is no edge between  $X$  and  $Z$ , meaning  $Y$  is a boundary between  $X$  and  $Z$ , then*

$$X \perp Z | Y \quad (68)$$

---

63. NDJE. see footnote This theory was initiated in Particle Physics Field Theory in 1954 by Murray Gell-Mann (1929-2019) and Francis E. Low (1921-2007) in the context of QED (Quantum Electrodynamics), and then generalized by Curtis Callan and Kurt Symanzik (1923-83) by establishing what are called the Callan–Symanzik equations. Developments in Statistical Mechanics date back to Kenneth G. Wilson's Ph.D. obtained under Gell-Mann's direction in 1961. Wilson made the connection with developments in Field Theory and developed the theory of critical exponents in connection with phase transitions, which became a prominent theme in the field in the 1970s, such as the famous "Les Houches Session XXVIII (1975): Methods in Field Theory" with remarkable contributions.

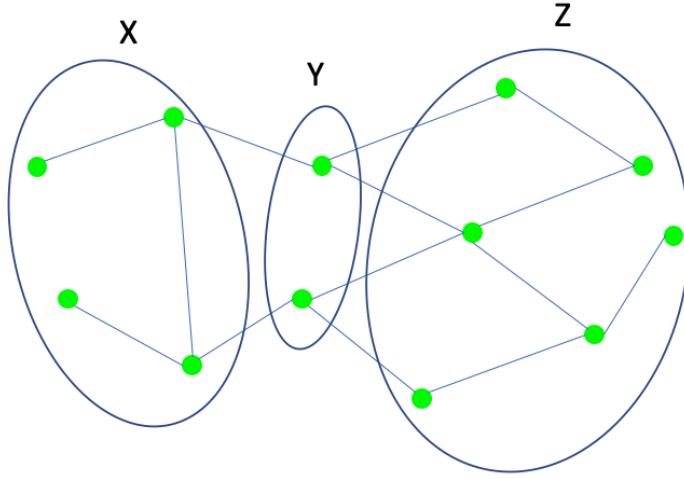


FIGURE 23 – Example of 3 variables ( $X, Y, Z$ ) composed of variables  $(x_i)_i$  from a graph  $G$  (undirected).

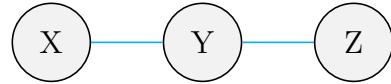


FIGURE 24 – Decomposition of a graph into 3 blocks ( $X, Y, Z$ ) with undirected connections.

such that

$$p(X, Z|Y) = \frac{p(X, Y, Z)}{p(Y)} = \frac{f_1(X, Y)}{\sqrt{p(Y)}} \frac{f_2(Y, Z)}{\sqrt{p(Y)}} = p(X|Y) p(Z|Y) \quad (69)$$

The proof follows the same approach as that adopted for the Markov property (Prop. 1) by considering the decomposition into cliques with  $x_i \in X$  and  $x_j \in Z$  and the  $x_k \in Y$ .

The converse is as follows:

**Property 3**

If in a graph  $G$ , there exist blocks of variables  $X, Y, Z$  such that

$$X \perp Z|Y \quad (70)$$

then we can write the probability distribution in the form of a graph whose morphology is that of Fig. 24. That is, there is no edge directly connecting variables from  $X$  to variables from  $Z$ .

When writing the joint probability, it decomposes easily using the conditional orthogonality property:

$$\begin{aligned} p(X, Y, Z) &= p(X, Z|Y)p(Y) = p(X|Y)p(Z|Y)p(Y) \\ &= p(X|Y)\sqrt{p(Y)} \times p(Z|Y)\sqrt{p(Y)} = f_1(X, Y) \times f_2(Z, Y) \end{aligned} \quad (71)$$

which indicates that the desired factorization is achieved. Thus, there is an equivalence between conditional independence and a factorization of the probability distribution into factors that do not directly relate variables. Note that blocks  $X$  and  $Z$  are not independent, i.e.,  $p(X, Z) \neq p(X)p(Z)$ , and  $Y$  **plays the role of a hidden variable**. Be aware that with such types of graphs, we cannot account for the situation of the graph in Fig. 25 where  $X$  and  $Z$  are independent but induce  $Y$  (there is causality). **Thus, Markov graphs do not exhaust all possibilities of independence.**

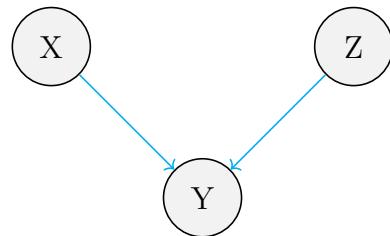


FIGURE 25 – Decomposition of a graph into 3 blocks  $(X, Y, Z)$  but with directional connections.

The generalization of the equivalence we just saw on 3 groups  $(X, Y, Z)$  is the result

of the **Hammersley-Clifford theorem**. First, let's state a definition.

**Definition 2** We say that a graph  $G$  is an I-map (Information map) if all conditional independence properties are in the topology of  $G$ <sup>a</sup>.

a. NDJE. see footnote 63 i.e., conditional independences between blocks are reflected in the connections between nodes.

**Theorem 3 (Hammersley-Clifford)**

Let a positive probability  $p(x_1, \dots, x_d) > 0$  where the  $x_i$  are random variables taking values in  $\chi$ , and let  $G$  be a non-directional graph, then  $G$  is an I-map on  $p$  iff  $p$  is a Gibbs distribution on the graph.

For the proof, it is first necessary to show that we can write  $p(x_1, \dots, x_d)$  in the form of a product of factors over cliques (cf. Markov field), and give it in the form of conditional independences between blocks of variables. It has been demonstrated that if  $p(x)$  can be expressed as a product of factors, then there are conditional independences between certain blocks.

The converse requires proving that if we have a set of conditions of the form " $X \perp Z|Y$ ", then we can construct cliques on  $G$  such that the probability distribution factorizes into products according to the definition of a Markov field (Def. 1). This was demonstrated in the case of 3 groups (Prop. 3). Here is the principle of generalization. For each node in the graph ( $x_i$  fixed), we consider the set of its neighbors (i.e., the nodes connected to it)

$$Y_i = \{x_j \in G \mid x_j \text{ is connected to } x_i\}$$

By construction,  $Y_i$  is the boundary between  $x_i$  and all other nodes  $x_k \notin Y_i$ . Therefore, we can write that

$$p(x_1, \dots, x_d) \propto \Phi_i^+(x_i, Y_i) \Phi_i^-(G \setminus \{x_i\})$$

where  $G \setminus \{x_i\}$  is the set of nodes in  $G$  different from  $x_i$ . This type of decomposition can be done for any  $x_i$ . Then, all these factorizations must be combined to obtain a minimal factorization. This process corresponds to taking intersections between constraints.

Let's see an example to relate this to the concepts of multi-scales: the **1D Brownian**

**motion** (random walk<sup>64</sup>). Let  $x_0 \sim \mathcal{N}(0, \sigma^2)$  for example, and the process that moves  $x_n$  to  $x_{n+1}$  ( $n \in \mathbb{N}$ ) is

$$x_{n+1} = x_n + z_{n+1}, \quad z_n \stackrel{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2) \quad (72)$$

We have  $x_n = x_0 + z_1 + \dots + z_n$ , thus

$$\begin{aligned} \mathbb{E}(x_n) &= 0 & \mathbb{E}(z_i z_j) &= \mathbb{E}(z_i) \mathbb{E}(z_j) = 0 \\ \mathbb{E}(x_n^2) &= (n+1)\sigma^2 & \mathbb{E}(x_n x_{n+k}) &= (n+1)\sigma^2 \end{aligned} \quad (73)$$

So, there is correlation at large distances.

Now, if we consider the sequence  $(x_0, x_1, x_2, x_3, \dots)$ , we want to construct a 2-to-2 dependency tree. For this, let's take  $(x_0, x_1)$  and build 2 new variables:

$$x_0^+ = \frac{x_0 + x_1}{\sqrt{2}} \quad x_0^- = \frac{x_0 - x_1}{\sqrt{2}} \quad (74)$$

and do the same for  $(x_2, x_3)$ ,  $(x_4, x_5)$ , etc. All pairs  $(x_{2k}, x_{2k+1})$  ( $k = 0, 1, \dots, n/2$ ) are thus transformed into pairs  $(+, -)$ . All variables of type  $"-"$  are independent of each other, but they are connected to variables of type "+". These variables of type +" can again be grouped 2-to-2: e.g.,  $(x_{2k}^+, x_{2k+2}^+)$  and proceed as before by performing their sum and their difference for example:

$$x_0^{+\pm} = \frac{x_0^+ \pm x_2^+}{\sqrt{2}} \quad x_4^{+\pm} = \frac{x_4^+ \pm x_6^+}{\sqrt{2}} \quad (75)$$

Now,  $x_0^{+-} = f(z_0, z_1, z_2)$  and  $x_4^{+-} = f(z_4, z_5, z_6)$  thus they are independent. And this holds for all  $x_{2p}^{+-}$ . Notice that  $x_0^{+-}$  depends on  $x_0^-$  and  $x_2^-$ , similarly  $x_4^{+-}$  depends on  $x_4^-$  and  $x_6^-$ . And so on, at each iteration, the variables of type  $x_i^{++\dots+}$  are orthogonalized, with the differences being independent of each other but dependent on two differences from the previous step. Thus, we have a cascade of orthogonal transformations with dependencies as illustrated in Fig. 26.

Thus, we have represented the **long-range interdependencies** among the variables  $x_i$  **through local but hierarchical dependencies**. Regarding the 1D random walk, we can simplify as we will see in the upcoming lecture. In the case of an image, this hierarchization

---

64. NDJE. See Course 2023 Sec. 6.4.2.2

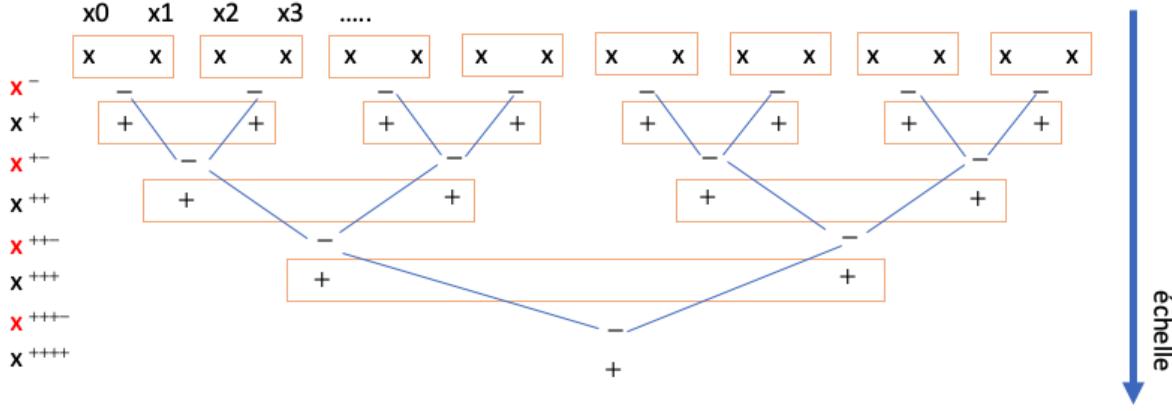


FIGURE 26 – Schematic representation of a cascade of orthogonal transformations applied to the initial variables  $x_i$  (from the random walk) and then to the variables of type  $x^{++\cdots+}$ . The variables  $x^{+\cdots-}$  are independent of each other at a given step, but depend on those from the previous step pairwise.

works very well, explains S. Mallat. Similarly, he outlines cases where connections can be made between variables obtained at the same level of the cascade. **Thus, we can use graphs that ultimately reflect complex situations that actually don't have so many connections.** The relationships between people in a company can be modeled in this type of graph.

In the case of the 1D random walk above, the cascade of transformations Eq. 74 corresponds to the Haar transform<sup>65</sup>. For example, if we consider the variable  $x_0^{++-}$ , we can view it from two perspectives which are applications of the Haar wavelet at two scales (Fig. 27)

$$x_0^{++-} = \frac{x_0^{++} - x_4^{++}}{\sqrt{2}} = \frac{(x_0 + x_1 + x_2 + x_3) - (x_4 + x_5 + x_6 + x_7)}{2^{3/2}} \quad (76)$$

We finally obtain an orthogonal representation of the original series of  $x_i$ , and while all the variables  $x_i$  interact, considering the variables at different scales, the interdependencies are only local. The Markov field is no longer on a single "purely temporal" axis as

65. NDJE. See Course 2018 Sec. 6.3 and Course 2021 Sec. 7.3.1

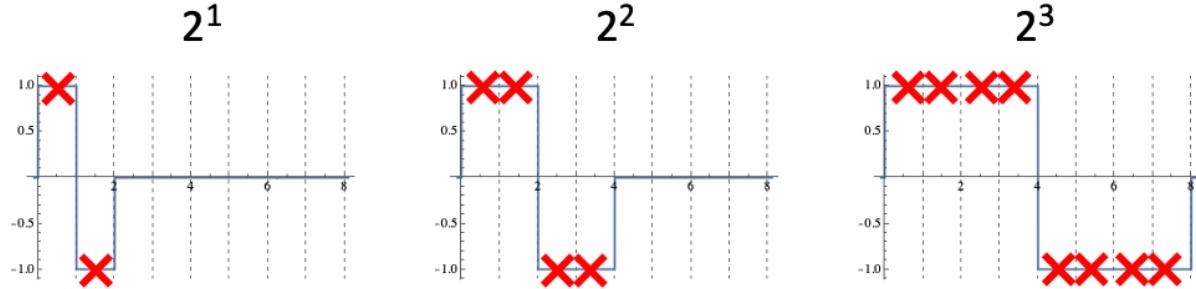


FIGURE 27 – Haar wavelet at different scales  $\Psi(x/2^n)$ . It is implicitly used in the construction of variables  $x^{++\dots}$  as in Equation 76 by reading the values given by the 'red crosses' at a given scale affected by the coefficient  $1/2^{2/n}$ .

in the 1D random walk (or in the case of the  $\phi^4$  theory field), but **there is a new axis that appears, which is the one of scales that allows transforming the dependencies by preserving only short-range interactions**. Such schemes can be applied in many applications of neural networks.

It is clear, says S. Mallat, that if large generative models are learnable, there must fundamentally be a simplification at play. Otherwise, if there is no source of simplification, a neural network cannot work, because of the curse of dimensionality that would require an exponential number of data points. **So, the real challenge is to find these sources of simplification.** That is, to find **representations of the data** that will simplify the pattern of interdependence, reducing it to **interactions of limited extent**. The example of the random walk is actually very generic because it brings out **the notion of scale which is ubiquitous** in the world around us.

Finally, all the optimization part with maximum likelihood is postponed to the next session, and we will enter the sampling problem. Indeed, while it is natural to conceive of Markov field graphs, sampling probabilities is more complex, as we need to estimate the normalization constant which is an integral in very high dimension. We will then see how Markov chains<sup>66</sup> help us.

---

66. NDJE. See Course 2023 Sec. 6.4.

## 5. Lecture of Feb. 7th

### 5.1 Introduction

S. Mallat invites us to watch recorded seminars from February 1-2, 2024, which he gave at ICTP in Trieste in front of a group of doctoral students and senior theoretical physicists. The recordings are available on his website<sup>67</sup>. This serves as a complement to the courses, providing insight into current research trends.

The link between Physics and Statistical Learning is emphasized in terms of the *emergence of structures*, which inform us about the interactions between the "atoms" of both domains. Physics has a model based on a much longer history. In the 1970s, whether in Statistical Physics or Particle Physics, the Renormalization Group Theory (see footnote 63) helps understand the evolution of structures across scales. In fact, this idea of *hierarchical* structures is found in many problems. S. Mallat recalls economist Herbert Simon's article<sup>68</sup>, which proposes a thesis that if organizations persist, it is because there is a hierarchical structure that ensures stability. However, while reading such an article is intellectually enriching, there is no underlying mathematical framework<sup>69</sup>.

In the previous session, we saw that in interaction trees, we can model not only horizontal, local interactions (hence the use of Markov fields) but also hierarchical interactions (scale axis) that can account for long-range interaction phenomena. The *Renormalization Group* works very well in Physics, but in Statistical Learning, we are faced with more complex problems, and in this context, we see that neural networks work very well. The question is Why? **This is why it is interesting to bridge what we can do in Physics on problems that are not so simple, and what we can do with neural networks on much more complicated problems like those we discussed at the beginning of this year's course.** Understanding these connections is a challenge in current research.

In a framework of images or time series where one can invoke *translation invariance* (e.g., recognizing a glass in a photo that can be taken from any viewpoint), we can outline

---

67. NDJE. The videos are also available here [www.ictp.it/home/salam-distinguished-lectures-series](http://www.ictp.it/home/salam-distinguished-lectures-series)

68. NDJE. See Course 2020 Sec. 3.2

69. NDJE. Between the lines, S. Mallat refers to the articles of Fisher and Shannon, whose ideas persist to this day. They were the subject of the 2022 course.

the following scheme: *translation invariance* implies an underlying *group*, which implies a *Fourier transform*, and then all of **Harmonic Analysis**. That is, in this simple framework, there is a whole branch of mathematics on which we can rely to define the notion of *scale*.

When we tackle *generation problems*, we are dealing with problems of a different complexity. Techniques using **Score Diffusion** are currently performing well, and there is a need to bridge the gap between these techniques and those that we can analyze within a solid mathematical framework. Note that the notion of *diffusion* is related to that of **denoising**, which has been a studied problem since at least the 1940s with Norbert Wiener's (1894–1964) filter. Many techniques have been developed in the "classical" denoising framework since that time, but almost overnight, neural networks have significantly improved efficiency. Strangely, we have a plethora of techniques that we control perfectly on one hand, and on the other hand, a nonlinear system that works much better, but "the why" remains a subject. Without a doubt, as S. Mallat suggests, understanding the link between the two types of techniques is necessary. Regarding the structure of the networks at work, we find a multi-scale structure to some extent.

## 5.2 Parameter Learning

### 5.2.1 Fisher and Shannon Frameworks

We consider the framework where we have chosen a family of probability densities  $\{p_\theta\}_\theta$ . Here, we adopt a **Gibbs modeling** such that

$$p_\theta(x) = Z_\theta^{-1} e^{-U_\theta(x)} \quad (77)$$

where  $U_\theta(x)$  defines a sort of parameterized energy. In the case of a neural network,  $\theta$  represents the set of weights of the network. Therefore, what matters to us now is finding the "best"  $\theta$ , denoted  $\theta^*$ , to best approximate the true<sup>70</sup> distribution  $p(x)$ . The only *a priori* knowledge we have is that of a collection of data  $\{x_i\}_{i \leq n}$  that we assume to be drawn *iid* from  $p$ . The structure of  $U_\theta(x)$  is also part of our *a priori* on the problem.

We need to define what is meant by " $p_{\theta^*}$  best approximates  $p$ ", i.e., the **metric** that will serve us for **optimization**. This problem was posed by Ronald A. Fisher (1890–1962)

---

70. NDJE. Recall that here we make an assumption that  $p(x)$  exists.

in 1922, which was the subject of the 2022 lecture<sup>71</sup>. Fisher provides a solution in the form of **maximum likelihood**. The idea is to choose  $\theta^*$  such that the probability of  $p_{\theta^*}(x_i)$ , on samples  $x_i$ , is as large as possible. Thus, we want to maximize the average  $p_\theta(x)$  ( $x \sim p$ ) or  $\log p_\theta(x)$  (maximum likelihood principle, also denoted MLE), i.e.:

$$\theta^* = \operatorname{argmax}_{\theta} \mathbb{E}_{x \sim p}(\log p_\theta(x)) \quad (78)$$

Another way to do this is to reinterpret this scheme: it is a view from Claude Shannon's (1916-2001) **Information Theory** in 1948<sup>72</sup>, through the notion of **Entropy** and that of the Kullback-Leibler divergence defined as<sup>73</sup>

$$D_{KL}(p||q) = \int p(x) \log \frac{p(x)}{q(x)} dx \quad (79)$$

We can immediately see the connection with the MLE<sup>74</sup>

$$D_{KL}(p||p_\theta) = \underbrace{\mathbb{E}_p(\log p)}_{-\mathbb{H}(p)} - \mathbb{E}_p(\log p_\theta(x)) \quad (80)$$

where we introduce  $\mathbb{H}(p)$ , the entropy of  $p$ , which is independent of  $\theta$  by definition. So, **maximizing likelihood is equivalent to minimizing the Kullback-Leibler divergence**,

$$\theta^* = \operatorname{argmin}_{\theta} D_{KL}(p||p_\theta) \quad (81)$$

One might wonder why  $D_{KL}(p||p_\theta)$  is a "natural" quantity to consider for our problem (we have seen why Fisher introduces MLE)? Ultimately,  $D_{KL}$  supports a **notion of similarity** between the two probability distributions. Recall two fundamental properties<sup>75</sup>:

71. NDJE. See Lecture 2022 Sec. 5, and Fisher's article <https://doi.org/10.1098/rsta.1922.0009> available on the course website <https://www.di.ens.fr/~mallat/CoursCollege.html>.

72. C. E. Shannon, The Bell System Technical Journal, Vol. 27, pp. 379–423, 623–656, July, October. Also available on the course website.

73. NDJE. We keep the notation from previous years.

74. NDJE. For the notations, note that  $\mathbb{E}_p(f(x))$  implicitly implies  $x \sim p$ .

75. NDJE. Also see Lecture 2019 Sec. 7.2.3 for example.

#### Property 4 (Kullback-Leibler)

- $D_{KL}(p||q) \geq 0$ , and = 0 iff  $p = q$ ;
- if  $p(x) = \prod_{i=1}^d p_i(x_i)$  manifesting the independence of the random variables  $x_i$  ( $x = (x_1, \dots, x_d)$ ), similarly for  $q(x)$ , then

$$D_{KL}(p||q) = \sum_{i=1}^d D_{KL}(p_i||q_i) \quad (82)$$

Note in passing an interpretation in terms of coding. If we want to encode the distribution  $p$  with the distribution  $q$ , the code is not optimal, and the number of bits lost is given by the Kullback-Leibler divergence.

The proof<sup>76</sup> of the first property uses Jensen's inequality. The proof of the second property is straightforward by taking two variables  $(x_1, x_2)$  through direct calculation using the property of log to transform a product into a sum, and then generalizing to any number of variables.

This viewpoint of using  $D_{KL}$  is that of neural networks simply because we are trying to model a probability: e.g., in supervised learning, classification<sup>77</sup> can be interpreted as maximizing the conditional probability of the class given the sample  $x_i$ , and in unsupervised learning, we encounter the same type of problem.

So, we are still in Fisher's scheme, what has changed concerns the expression of  $p_\theta$ . It has transitioned from a simple Gaussian, in the original article, whose covariance depends on  $\theta$ , to more complex forms in neural networks.

#### 5.2.2 Optimization to find $\theta^*$

We need to maximize the likelihood  $\log p_\theta$  or minimize its opposite  $-\log p_\theta$ . Thus, we consider the cost function (*loss*)  $\ell(\theta)$

$$\ell(\theta) = -\mathbb{E}_{x \sim p}(\log p_\theta(x)) = \log Z(\theta) + \int p(x)U(x; \theta) dx \quad (83)$$

---

76. NDJE. This can be found, for example, in the 2022 lecture Th. 13.

77. NDJE. See, for example, Lecture 2019 Secs. 7.2.3 and 7.3.2.

if we denote  $U(x; \theta)$  as the argument of the exponent in Eq. 77. The tool to minimize  $\ell(\theta)$  is the **gradient descent**<sup>78</sup>. Under this term, there are many algorithms and implementations<sup>79</sup>. As preliminary, we have this result:

### Lemma 1

$$\nabla_{\theta} \ell(\bar{\theta}) = \mathbb{E}_{x \sim p} (\nabla_{\theta} U(x; \bar{\theta})) - \mathbb{E}_{x \sim p_{\bar{\theta}}} (\nabla_{\theta} U(x; \bar{\theta})) \quad (84)$$

The proof is simple, just remember that

$$Z(\theta) = \int e^{-U(x; \theta)} dx \quad (85)$$

which gives, when differentiating the first term of the definition in Eq. 83

$$\frac{\nabla_{\theta} Z(\bar{\theta})}{Z(\bar{\theta})} = - \int \nabla_{\theta} U(x; \bar{\theta}) p_{\bar{\theta}}(x) dx = -\mathbb{E}_{x \sim p_{\bar{\theta}}} (\nabla_{\theta} U(x; \bar{\theta})) \quad (86)$$

The immediate consequence of this lemma is that

$$\nabla_{\theta} \ell(\theta^*) = 0 \Leftrightarrow \mathbb{E}_{x \sim p} (\nabla_{\theta} U(x; \theta^*)) = \mathbb{E}_{x \sim p_{\theta^*}} (\nabla_{\theta} U(x; \theta^*)) \quad (87)$$

In general, a **moment** is called a quantity

### Definition 3 (**moment**)

$$\mathbb{E}_{x \sim p}(f(x)) = \int f(x)p(x) dx \quad (88)$$

In expression 87,  $f(x) = \nabla_{\theta} U(x; \theta^*)$  is a vector. So, we have equality between two (families) of moment vectors. **The moments**  $\mathbb{E}_{x \sim p} (\nabla_{\theta} U(x; \theta^*))$  **are constraints**<sup>80</sup> **that must be**

---

78. NDJE. See Lecture 2018 Sec. 10, Lecture 2019 Sec.4.2.2, Lecture 2022 Sec. 3.6.2

79. NDJE. See, for example, S. Ruder's article, "An overview of gradient descent optimization algorithms", <https://arxiv.org/abs/1609.04747>.

80. NDJE. In the 2023 Lecture concerning Gibbs' Theorem (Th. 17 Sec. 7.6), these are the expectations of functions  $\phi_k(x)$  ( $\mathbb{E}_{x \sim p}(\phi_k(x))$ ) that are the constraints to define  $p_{\theta}$ . In this case, it is shown that  $p_{\theta}$  takes the form of a Gibbs distribution.

**satisfied for the distribution**  $p_\theta$ . This is called a *moment projection*.

Now, the *gradient descent* as such is a way to gradually modify the value of  $\theta$  to obtain  $\theta^*$ . At step  $t + 1$ , knowing the values at iteration  $t$ , we have

$$\theta_{t+1} - \theta_t = -\varepsilon \nabla_\theta \ell(\theta_t) = \varepsilon \left( \mathbb{E}_{x \sim p_{\theta_t}} (\nabla_\theta U(x; \theta_t)) - \mathbb{E}_{x \sim p} (\nabla_\theta U(x; \theta_t)) \right) \quad (89)$$

We modify the value of  $\theta$  in the direction of the error of the moments. **The main problem of the method lies in the computation of these moments.**

Regarding the expectation with the distribution  $p$ , we use the initial data  $(x_i)_i$  which are assumed to be *iid* samples from  $p$ :

$$\mathbb{E}_{x \sim p} (\nabla_\theta U(x; \theta_t)) = \int \nabla_\theta U(x; \theta_t) p(x) dx \approx \frac{1}{n} \sum_{i=1}^n \nabla_\theta U(x_i; \theta_t) \quad (90)$$

However, concerning the expectation with the distribution  $p_{\theta_t}$ , **we need to construct *iid* samples**. MCMC methods will provide us with the means, but the implementation is not straightforward *a priori*. Note, however, that on the one hand, we must proceed to construct a chain long enough for the average to converge reasonably to the expectation, and on the other hand, this generation step must be repeated for each iteration of the gradient descent. **In other words, even if the method seems clear to implement, it can be very slow.** Historically, since roughly the works of Fisher and those of Shannon, the paradigm had not changed until 2005 when the idea of **score matching** emerged (Aapo Hyvärinen<sup>81</sup>). We will see why this technique accelerates the process, but there is no miracle, there will be a price to pay.

Now, the general paradigm of choosing an expression for  $p_\theta$ , using a function to minimize to find the best  $\theta$  by gradient descent is undoubtedly what we want to do, says S. Mallat. We may want to replace the Kullback-Leibler divergence with another function to minimize to speed up the process, but before that, we need to better understand the properties of this "metric".

---

81. NDJE. S. Mallat refers to the article Hyvarinen, A., (2005), "Estimation of non-normalized statistical models by score matching", Journal of Machine Learning Research, Vol 6(Apr), pp. 695–709. <https://jmlr.org/papers/volume6/hyvarinen05a/hyvarinen05a.pdf>.

### 5.3 The linear case

This is a very important particular case of the energy form such that<sup>82</sup>

$$U(x; \theta) = \sum_{k=1}^K \theta_k \phi_k(x) = \theta^T \Phi(x) \quad (91)$$

where  $\theta = \{\theta_k\}_{k \leq K}$  and the functions  $\Phi = \{\phi_k\}_{k \leq K}$  are known in advance. It is a linear decomposition on a linear space

$$U(x; \theta) \in \mathcal{V} = \text{Vect}\{\phi_k\}_{k \leq K} \quad (92)$$

This type of parameterization is typical of situations encountered in Physics, for example. Note that in Machine Learning,  $\Phi$  is a *feature vector*. The  $\theta_k$  can be seen as coupling coefficients.

In this linear framework, the gradient is easy to compute since:

$$\nabla_\theta U(x, \bar{\theta}) = \Phi(x) \quad (93)$$

and it is independent of  $\theta$ . The constraints given by the equality between moments (Eq. 87) translate as

$$\forall \theta, \quad \mathbb{E}_{x \sim p_\theta}(\Phi(x)) = \mathbb{E}_{x \sim p}(\Phi(x)) \quad (94)$$

Then we have the following property

**Theorem 4 (Maximum Entropy)**

Stating that  $D(p||p_{\theta^*})$  is minimum, i.e.,  $\theta^*$  is optimal (MLE), is equivalent to stating that  $p_{\theta^*}$  is the distribution that maximizes the entropy

$$\mathbb{H}(q) = -\mathbb{E}_q(\log q) = - \int p(x) \log p(x) \, dx \quad (95)$$

and such that

$$\mathbb{E}_{x \sim q}(\Phi(x)) = \mathbb{E}_{x \sim p}(\Phi(x)) \quad (96)$$

---

82. NDJE. See for example Course 2022 Sec 4.2, Course 2023 Sec. 7.6.

This theorem tells us that we can thus forget somewhat **the parametric problem**, by **substituting it with the search for a distribution constrained by moments and satisfying the Maximum Entropy Principle**<sup>83</sup>. It is a reinterpretation in terms of Information Theory of the parametric problem.

**Proof 4.** We have an optimization problem of a concave function ( $\mathbb{H}(q)$ ) where the unknown is  $q(x)$  satisfying the collection of equalities ( $K$  constraints)

$$\forall k, \quad \mathbb{E}_q(\phi_k(x)) = \int q(x)\phi_k(x) dx = \mathbb{E}_p(\phi_k(x)) = \mu_k \quad (97)$$

Moreover, since  $q$  is a probability,  $\int q(x)dx = 1$ . Now, this type of problem is solved by the Lagrange multipliers<sup>84</sup>. As  $\mathbb{H}(q)$  is strictly concave, the solution to the problem is unique.

So, consider the Lagrangian

$$\mathcal{L}(q, \theta) = \mathbb{H}(q) + \sum_{i=1}^K \theta_k c_k(q) + \theta_0 \left( \int q(x)dx - 1 \right) \quad (98)$$

with  $c_k(q) = \mu_k - \int q(x)\phi_k(x) dx$ . *NDJE. The proof is identical to that of Course 2023 Th. 17 (Gibbs)*, by requiring that the derivative of  $\mathcal{L}(q, \theta)$  with respect to the distribution  $q$  is zero, we then find that the optimal distribution is written as

$$p^*(x, \theta) = Z(\theta)^{-1} \exp \left\{ - \sum_{k=1}^K \theta_k \phi_k(x) \right\} \quad Z(\theta) = e^{\theta_0 - 1} = \int \exp \left\{ - \sum_{k=1}^K \theta_k \phi_k(x) \right\} dx \quad (99)$$

and the Lagrange multipliers  $\theta_k$  are such that

$$\int p^*(x, \theta) \phi_k(x) dx = \mu_k \quad (100)$$

we denote  $\theta^*$  their values. Furthermore, we show

$$D_{KL}(p||p_{\theta^*}) = \mathbb{H}(p_{\theta^*}) - \mathbb{H}(p) \quad (101)$$

---

83. NDJE. Regarding the Maximum Entropy Principle, see also Course 2022 Sec. 7.5

84. NDJE. See Course 2018 Sec. 8.3 Kuhn & Tucker

Indeed,

$$\begin{aligned}
D_{KL}(p||p_{\theta^*}) &= \int p(x) \log \frac{p(x)}{p_{\theta^*}(x)} dx \\
&= -\mathbb{H}(p) - \int p(x) \left( -\log Z(\theta^*) - \sum_k \phi_k(x) \theta_k^* \right) dx \\
&= -\mathbb{H}(p) + \log Z(\theta^*) + \sum_k \mathbb{E}_p(\phi_k(x)) \theta_k^*
\end{aligned} \tag{102}$$

Now, for the optimal distribution obtained above,  $\mathbb{E}_p(\phi_k(x)) = \mathbb{E}_{p_{\theta^*}}(\phi_k(x))$ , thus

$$\begin{aligned}
D_{KL}(p||p_{\theta^*}) &= -\mathbb{H}(p) - \underbrace{\int p_{\theta^*}(x) (-\log Z(\theta^*) - \sum_k \theta_k^* \phi_k(x)) dx}_{\log p_{\theta^*}} \\
&= \mathbb{H}(p_{\theta^*}) - \mathbb{H}(p)
\end{aligned} \tag{103}$$

■

So, from the constraints  $\mu_k$  estimated from  $\phi_k(x_i)$  where  $x_i$  are the initial samples, we impose the criterion of maximum entropy to find the distribution  $p_{\theta^*}$ . Maximum entropy ensures the uniformization of the distribution over typical sets<sup>85</sup>. In a way, we do not favor particular samples/particular configurations in these sets. In Statistical Physics, one would invoke a Boltzmann microcanonical ensemble. In this context, **the Kullback-Leibler divergence is a measure of the excess entropy of the distribution  $p_{\theta^*}$**  because  $D_{KL}(p||q) \geq 0$ . If there is too much entropy (i.e.,  $D_{KL}(p||q) > 0$ ), the question that arises is: **what is the right representation of  $\Phi$ ? The best model is the one with the smallest entropy, while still satisfying all the constraints and the Maximum Entropy Principle.**

## 5.4 Score Matching

During gradient descent (Eq 89), we saw that at each step  $t$  we need to compute the expectation  $\mathbb{E}_{x \sim p_{\theta_t}}(\nabla_{\theta} U(x; \theta_t))$ , which is very costly because it requires sampling from the distribution  $p_{\theta_t}$ . But if we go back to the expression of the loss (Eq. 83) that we rewrite

---

<sup>85</sup> NDJE. Course 2022 Sec. 7.5

here for convenience:

$$\ell(\theta) = -\mathbb{E}_{x \sim p}(\log p_\theta(x)) = \log Z(\theta) + \int p(x)U(x; \theta) dx$$

we know that during gradient calculation, the second term is not a problem because it involves  $\mathbb{E}_{x \sim p}(\nabla_\theta U(x; \theta_t))$ , which can be approximated with the average over samples  $(x_i)_i$ . So the problem actually lies in calculating the gradient of the 1st term, which gives us an expectation with the probability  $p_{\theta_t}$  and is difficult to sample. **Therefore, everything would be simpler if we didn't have to deal with a normalization constant.** Because it involves a high-dimensional integral and we need to invoke MCMC again. How to overcome this?

We can notice that

$$D_{KL}(p||p_\theta) = \int p(x) (\log p(x) - \log p_\theta(x)) dx \quad (104)$$

which shows the difference of log. To avoid  $Z(\theta)$  in the second term, it "suffices" to consider the gradient with respect to  $x$ , and ensure we have a positive quantity to minimize. Thus, we look at the following quantity

$$I(p, p_\theta) = \frac{1}{2} \int p(x) \|\nabla_x \log p(x) - \nabla_x \log p_\theta(x)\|^2 dx \quad (105)$$

which uses

$$\nabla_x \log p_\theta(x) = \nabla_x(-\log Z(\theta) - U(x; \theta)) = -\nabla_x U(x; \theta) \quad (106)$$

and makes  $Z(\theta)$  disappear. The quantity  $I(p, p_\theta)$  was introduced by R. Fisher (already in 1923), it is **the Fisher divergence**. *NDJE. Note: the score involved in the Fisher Information invokes  $\nabla_\theta \log p_\theta(x)$  (Course 2022 Sec. 5.3). But here note  $\nabla_x$  of log of probabilities. There is a historical reason<sup>86</sup> for which both formulations are called scores, but one must*

<sup>86</sup> NDJE. dee footnote The Fisher Information is defined (1D) as

$$I(\theta) = \mathbb{E}_{p_\theta} \left[ \left( \frac{\partial \log p_\theta(x)}{\partial \theta} \right)^2 \right] = \int_{\mathbb{R}} p_\theta(x) \left( \frac{\partial \log p_\theta(x)}{\partial \theta} \right)^2 dx$$

with  $p_\theta(x)$  a pdf parameterized by  $\theta$ . From any pdf  $p(x)$ , we can define a parametric form  $p(x - \theta)$ .

be careful.

**So, Score Matching consists of minimizing  $I(p, p_\theta)$  instead of  $D_{KL}(p||p_\theta)$ .** The big advantage is that this is much easier to compute. However,  $D_{KL}(p||p_\theta)$  is the "natural metric", which made it the *a priori* choice. In fact, the idea of using the Fisher divergence came to the forefront because the community was faced with practical computation problems in high dimensions. This metric is used in large generative models.

We need to study the link between  $D_{KL}(p||p_\theta)$  and  $I(p, p_\theta)$ , which is not *a priori* the optimal choice of metric. The questions that arise are: do we obtain the same distribution  $p_{\theta^*}$ ? and will it converge with a reasonable number of samples? It is this second question that is the most critical, in fact. In this context, the constants of *log-Sobolev* will appear, which establish a connection with a chapter in Analysis that studies equivalences between metrics on functions.

## 5.5 Study of an example: $\phi^4$ potential

We revisit the example taken from Physics seen in section 4.2.2.2. For memory, the energy term takes the form of the following Eq. 66:

$$U(x) = \frac{\beta}{2} \sum_{(i,j)} (x(i) - x(j))^2 + \sum_{i=1}^N V(x(i)), \quad p(x) \propto e^{-U(x)} \quad (107)$$

which contains a kinetic energy term (note: discretization of the Laplacian after integration by parts and  $(i, j)$  denotes pairs of nearest neighbor spins on an  $N \times N$  grid) and a potential term which for the  $\phi^4$  theory is shaped like 2 attractive wells  $x = \pm 1$  ( $V(t) = (t^2 - 1)^2$ ).

Deriving with respect to  $\theta$  or with respect to  $x$  is equivalent. Then

$$I(\theta) = \int_{\mathbb{R}} p(x - \theta) \left( \frac{\partial \log p_\theta(x - \theta)}{\partial \theta} \right)^2 dx$$

and, by change of variable, this quantity is independent of  $\theta$  and can be written as

$$I[p] = \int_{\mathbb{R}} p(x) \left( \frac{\partial \log p(x)}{\partial x} \right)^2 dx$$

which can be interpreted as the Fisher Information of the probability distribution  $p$ . The notion of divergence follows.

We want to find the underlying distribution while only having examples  $\{x_i\}_{i \leq n}$  (note: here, the index  $i$  denotes the sample  $x_i$ , and  $x_i(k)$  denotes the pixel/spin  $k$  of the sample). We do not know *a priori* the form of 107 for the energy. This situation is typical of problems in Statistical Learning. It differs from Physics, where the problem has been studied for a long time to have an idea of  $U(x)$ . So, what to do? Based on what we have seen in the previous sections, we define a model, i.e., here *a family of energies*  $U_\theta(x)$ . Let's say we have *a priori information* that there are two terms:

$$U_\theta(x) = \frac{1}{2}x^T K x + \sum_i V_\gamma(x(i)) \quad (108)$$

the first is the form of a *Gaussian model*, and the second is a form of a *parameterized scalar potential* ( $\gamma$ ). If we impose stationarity, thus invariance by translation, then  $K$  is a *convolution operator*<sup>87</sup> with finite support:

$$\begin{aligned} x^T K x &= \sum_i x(i)(K * x)(i) = \sum_{i,j} x(i)K(i-j)x(j) \\ &= \sum_{i,\tau} x(i)K(\tau)x(i-\tau) = \sum_\tau K(\tau) \sum_i x(i)x(i-\tau) \\ &= \sum_\tau K(\tau)(x * \tilde{x})(\tau) = K^T(x * \tilde{x}) \end{aligned} \quad (109)$$

where  $\tilde{x}(i) = x(-i)$ . We do not really know the support, so we need to determine  $K(\tau)$  for  $|\tau| < C$  with  $C$  unknown. Regarding the potential term whose form is unknown, we can only invoke a parameterization in a basis of functions (e.g., piecewise linear functions):

$$V_\gamma(t) = \sum_{k=1}^K \gamma_k \rho_k(t) = \gamma^T \rho(t) \quad (110)$$

with vectors  $\gamma = (\gamma_k)_k$  and  $\rho = (\rho_k)_k$ . We can for example take  $\rho_k(t)$  as a linear rectifier (ReLU) translated by a factor depending on  $k$ . Indeed, with 2 ReLUs we can construct a triangle function (Fig. 28) at the basis of the decomposition into piecewise linear functions as a consequence of the sampling theorem<sup>88</sup>. However, we can also take linear combinations of polynomials, etc. We can write the term of energy corresponding to the scalar

87. NDJE. See Course 2023 Sec. 9.3

88. NDJE. Course 2018 Sec. 5.2.4

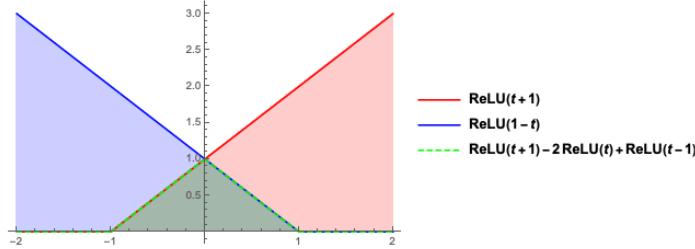


FIGURE 28 – With a linear combination of ReLUs, we can obtain a "triangle" function (in green) (*NB: excerpt from Course 2019 Sec. 5.3.2*).

potential as:

$$\sum_i V_\gamma(x(i)) = \gamma^T \sum_i \rho(x(i)) = \gamma^T \Gamma(x) \quad (111)$$

where  $\Gamma_k(x) = \sum_i \rho_k(x(i))$ . Thus, the energy can be written according to the following expression:

$$U_\theta(x) = \frac{1}{2} K^T(x * \tilde{x}) + \gamma^T \Gamma(x) = \Theta^T \Phi(x) \quad (112)$$

where we identify the parameters  $\theta = (K, \gamma)$  and the *feature vector*  $\Phi(x) = \{1/2(x * \tilde{x}), \Gamma(x)\}$ . If we think of  $x$  as an image, the first term is the convolution of the image with itself, while the second term can be the average of ReLUs over the entire image.

Now that we have identified the parametric expression of the energy  $U_\theta(x)$ , we can unfold the formalism described in the previous sections. That is, find  $\theta$  such that  $\mathbb{E}_{p_\theta}(\Phi(x))$  equals the measures  $\mathbb{E}_p(\Phi(x))$  estimated with the data  $(x_i)_i$ . We use gradient descent with a metric (Kullback-Leibler or Fisher divergence). In the end, we have a probability distribution model describing the data. What we want to do next is to generate new samples  $x$  *iid* according to  $p_{\theta^*}$ . For this, we need to do sampling.

Before that, we need to understand the difference between the metrics.

## 6. Lecture of February 14th

We will study in more detail the differences between the **Kullback-Leibler divergence**, which is the natural "metric" for estimating the best approximation of the data

distribution ( $p(x)$ ), and, for technical reasons, especially in large generative models, we have come to use **Fisher divergence** (notion of **Score matching**). For example, we need to understand when minimizing the score gives us a good approximation of  $p(x)$ ?

## 6.1 Total variation distance

As we have seen in Section 5.2.1, determining the optimal parameter set  $\theta^*$  is done by minimizing the divergence  $D_{KL}(p||p_\theta)$  which describes the similarity between the data distribution  $p(x)$ , and an element  $p_\theta(x)$  of a parameterized family. Thus,

$$\theta^* = \operatorname{argmin}_\theta D_{KL}(p||p_\theta) \quad (113)$$

We had seen two properties of this divergence (Prop. 4). We will give another one related to the total variation distance<sup>89</sup>

### Definition 4 (Total variation distance)

Let  $p, q$  be two probability densities on  $\mathcal{E}$  a measurable set (let's say  $\mathbb{R}^d$ ), we define the total variation distance as follows

$$d_{TV}(p, q) = 2 \sup_{\mathcal{A} \subset \mathcal{E}} |p(\mathcal{A}) - q(\mathcal{A})| = \int_{\mathcal{E}} |p(x) - q(x)| dx = \|p - q\|_{L1} \leq 2 \quad (114)$$

with  $d_{TV}(p, q) = 2$  if the supports of  $p$  and  $q$  are disjoint.

The total variation distance is defined in a broader framework of probability measure theory, henceforth we will denote it by  $\|p - q\|_{TV}$ .

To prove the equality between the two definitions, we can proceed as follows. Let the set

$$\mathcal{A}_+ = \{x / p(x) > q(x)\} \quad (115)$$

and its complement in  $\mathcal{E}$

$$\mathcal{A}_+^c = \{x / p(x) \leq q(x)\} \quad (116)$$

---

89. NDJE. I have specialized it for *pdfs* but it is defined for measures more broadly.

For  $\mathcal{A}_+ \subset \mathcal{E}$  arbitrary<sup>90</sup>, we have the following double inequality

$$m = p(\mathcal{A}_+^c) - q(\mathcal{A}_+^c) \leq p(A) - q(A) \leq p(\mathcal{A}_+) - q(\mathcal{A}_+) = M \quad (117)$$

We can prove, for example, the second inequality as follows:

$$\begin{aligned} p(\mathcal{A}) - q(\mathcal{A}) &= \underbrace{p(\mathcal{A} \cap \mathcal{A}_+) - q(\mathcal{A} \cap \mathcal{A}_+)}_{>0} + \underbrace{p(\mathcal{A} \cap \mathcal{A}_+^c) - q(\mathcal{A} \cap \mathcal{A}_+^c)}_{\leq 0} \\ &\leq p(\mathcal{A} \cap \mathcal{A}_+) - q(\mathcal{A} \cap \mathcal{A}_+) \\ &\leq p(\mathcal{A}_+) - q(\mathcal{A}_+) \end{aligned} \quad (118)$$

The proof of the first inequality proceeds in a similar manner. Now, we can notice that  $\mathcal{A}_+ \subset \mathcal{E}$ , so the supremum is achieved for  $\mathcal{A} = \mathcal{A}_+$ , thus

$$\|p - q\|_{TV} = 2|p(\mathcal{A}_+) - q(\mathcal{A}_+)| = 2M \quad (119)$$

Now  $p(\mathcal{E}) = q(\mathcal{E}) = 1$ , so  $m + M = p(\mathcal{E}) - q(\mathcal{E}) = 0$  hence  $m = -M$ . Thus

$$\|p - q\|_{TV} = 2M = M - m = \int_{\mathcal{A}_+} |p(x) - q(x)| dx + \int_{\mathcal{A}_+^c} |p(x) - q(x)| dx = \int_{\mathcal{E}} |p(x) - q(x)| dx \quad (120)$$

The divergence  $D_{KL}(p||q)$  controls the value of this distance according to the following inequality established by Mark Semenovich Pinsker (1925-2003):

### Theorem 5 (Pinsker's inequality)

$$\|p - q\|_{TV}^2 \leq 2D_{KL}(p||q) \quad (121)$$

### Proof 5.

The proof follows a typical pattern in this field. Let's consider the case of probability densities, and forget in the notations about the set  $\mathcal{E}$  on which the integrals are performed. We also place ourselves under the same conditions of validity of  $D_{KL}(p||q)$ . Thus,

$$\|p - q\|_{TV} = \int |p(x) - q(x)| dx = \int q(x) \left| \frac{p(x)}{q(x)} - 1 \right| dx = \mathbb{E}_q[|f(x)|] \quad (122)$$

---

90. NDJE: I simplify the notations, one should consider Borel sets of  $\mathcal{E}$ , and we use the Lebesgue measure.

if  $f(x) = \frac{p(x)}{q(x)} - 1 \geq -1$ . Note in passing that  $\mathbb{E}_q(f(x)) = 0$ . On the other hand, we can write

$$\begin{aligned} D_{KL}(p||q) &= \int p(x) \log \frac{p(x)}{q(x)} dx = \mathbb{E}_q[(1 + f(x)) \log(1 + f(x))] \\ &= \mathbb{E}_q[(1 + f(x)) \log(1 + f(x)) - f(x)] \end{aligned} \quad (123)$$

We can show<sup>91</sup> furthermore that  $\forall y > -1$ ,

$$(1 + y) \log(1 + y) - y \geq \frac{1}{2} \frac{y^2}{1 + y/3} \quad (124)$$

Thus

$$D_{KL}(p||q) \geq \frac{1}{2} \mathbb{E}_q \left[ \frac{f(x)^2}{1 + f(x)/3} \right] \quad (125)$$

To conclude, we use the following lemma<sup>92</sup> on the rvs  $X$  and  $Y$ , assuming that  $Y \geq 0$

$$\mathbb{E}(|X|^2/Y) \geq \frac{\mathbb{E}(|X|)^2}{\mathbb{E}(Y)} \quad (126)$$

Finally, we arrive at the relation

$$D_{KL}(p||q) \geq \frac{1}{2} \frac{\mathbb{E}_q[|f(x)|]^2}{\mathbb{E}_q[1 + f(x)/3]} = \frac{1}{2} \mathbb{E}_q[|f(x)|]^2 = \frac{1}{2} \|p - q\|_{TV}^2 \quad (127)$$

which is what we wanted to prove. ■

This theorem tells us that **the Kullback-Leibler divergence** is indeed asymmetric and not a distance, yet it **nevertheless controls a total variation distance that is quite strong**. In particular, minimizing  $D_{KL}(p||q)$  effectively ensures that two probability distributions will converge to each other.

---

91. NDJE. Hint: initially, we can expand both functions in a series around  $y = 0$  to understand the choice of function on the right-hand side of the inequality. Then by defining a function  $f$  as the difference between the function on the left-hand side and the one on the right-hand side, by taking the second derivative of  $f$ , we find that  $f''$  is strictly positive on  $(-1, \infty)$ . The function  $f'$  is strictly increasing from  $-\infty$  to  $\infty$  and its only zero is at  $y = 0$ . The function  $f$  is therefore strictly decreasing on  $(-1, 0]$  and strictly increasing on  $[0, \infty)$  with its minimum at  $y = 0$  where it is equal to 0.

92. NDJE. Use Cauchy-Schwartz with the two rvs  $|X|/\sqrt{Y}$  and  $\sqrt{Y}$ .

## 6.2 The Hessian of $D_{KL}$ in the Linear Case

Again, our objective is to find the best distribution  $p_\theta(x)$  approximating the data distribution (denoted  $p(x)$ ), and we have identified in Section 5.2.1 that the natural "metric" is  $D_{KL}(p||p_\theta)$ . Furthermore, according to equations 80 and 83, we have

$$D_{KL}(p||p_\theta) = \mathbb{H}[p] + \ell(\theta) \quad (128)$$

highlighting the log-likelihood<sup>93</sup>

$$\ell(\theta) = -\mathbb{E}_p(\log p_\theta(x)) \quad (129)$$

Minimization by gradient descent has allowed us to establish Lemma 1, such that the condition  $\nabla_\theta \ell(\theta) = 0$  gives

$$\mathbb{E}_{x \sim p}(\nabla_\theta U(x; \theta^*)) = \mathbb{E}_{x \sim p_{\theta^*}}(\nabla_\theta U(x; \theta^*)) \quad (130)$$

This means that the best distribution  $p_\theta$  is the one that accounts for the moments here defined as the gradient of the energy with respect to  $\theta$ . The computational problem lies in the right term, as we highlighted at the end of Section 5.2.2. And in Section 5.4, we identified the real challenge, namely the computation of the normalization constant  $Z(\theta)$ . This is what motivates the use of *score matching*.

But before we see that, does gradient descent converge? We saw, in the exponential case where the energy has a linear expression (Sec. 5.3) of the form

$$U(x; \theta) = \theta^T \Phi(x) \quad (131)$$

where  $\Phi(x)$  is a features vector, that

$$\nabla_\theta D_{KL}(p||p_\theta) = \mathbb{E}_p(\Phi(x)) - \mathbb{E}_{p_\theta}(\Phi(x)) \quad (132)$$

However, if we want an easily solvable problem with gradient descent, it is desirable to be in a convex case, i.e., we need to study the Hessian. So let's state the following theorem:

---

93. NDJE. Note the sign difference compared to the definition of the loss I took.

**Theorem 6**

The Hessian with respect to  $\theta$  of  $D_{KL}(p||p_\theta)$  in a case where the energy has a linear form  $U(x; \theta) = \theta^T \Phi(x)$  ( $\Phi = \{\phi_k\}_{k \leq K}$ ) is such that

$$H_\theta[D_{KL}(p||p_\theta)] = \text{Cov}_{p_\theta}(\Phi(x)) = \mathbb{E}_{p_\theta}(\Phi(x)\Phi(x)^T) - \mathbb{E}_{p_\theta}(\Phi(x))\mathbb{E}_{p_\theta}(\Phi(x)^T) \geq 0 \quad (133)$$

The demonstration is straightforward by considering the expression of the Hessian  $H_\theta = \nabla_\theta \nabla_\theta^T$  and using the expression Eq. 86. Therefore, since the Hessian is always positive, we know that the problem is simpler, and gradient descent converges to the solution. However, two points emerge:

- as already mentioned, there is a problem with computing  $Z(\theta)$ , which requires a large number of samples;
- the Hessian can be poorly conditioned, meaning that the extreme eigenvalues can be very different<sup>94</sup>, which affects the convergence speed.

Despite these two challenges, this approach has been used and was only recently replaced by score matching. The method worked well because it dealt with relatively low-dimensional problems. The problem becomes more acute as the dimensionality increases.

### 6.3 Fisher Divergence (Score Matching)

The solution found in 2005 by Aapo Johannes Hyvärinen (1970-) (see footnote 81) is to use a new metric (Sec. 5.4). We will replace

$$D_{KL}(p||p_\theta) = \mathbb{E}_p[\log p(x) - \log q(x)] \quad (134)$$

with the following quantity called **Fisher divergence** in the literature, related to the Fisher Information (see footnote 86)

$$I(p, p_\theta) = \frac{1}{2}\mathbb{E}_p[\|\nabla_x \log p(x) - \nabla_x \log p_\theta(x)\|^2] \quad (135)$$

---

94. NDJE. See Course 2019 Sec. 8.2.2

which, as we have seen, allows us to **get rid of the constant  $Z(\theta)$  because we take the gradient with respect to  $x$** , and the norm ensures positivity. So, the idea is to minimize  $I(p, p_\theta)$  by gradient descent:

$$\theta_{SM} = \operatorname{argmin}_\theta I(p, p_\theta) \quad (136)$$

( $\theta_{SM}$  is then the  $\theta$  of the score matching). The first remark is that the calculation of the gradient descent is easier, but the second remark concerns why is this method as precise as the one obtained with  $D_{KL}(p||p_\theta)$ ? This latter point is more delicate. Because, one can find oneself in a situation where the method using Fisher divergence is much slower, and thus we are faced with the problem of controlling one metric by another. This branch of Analysis (e.g., the log-Sobolev constants) has been particularly developed in the last 30 years, S. Mallat tells us.

## 6.4 Rewriting of $I(p, p_\theta)$

Let us first study the easy part of the convergence by this theorem:

### Theorem 7 (Hyvärinen-2005)

*Under the regularity assumption  $\sup_x \|\nabla_x U_\theta(x)\| < \infty$ , we can rewrite  $I(p, p_\theta)$  as follows:*

$$I(p, p_\theta) = \mathbb{E}_p \left[ \Delta_x U_\theta(x) + \frac{1}{2} \|\nabla_x U_\theta(x)\|^2 \right] + C(p) = J(\theta) + C(p) \quad (137)$$

*where  $C(p)$  is a constant depending on the distribution  $p$ .*

We see the potential gain in the fact that  $I(p, p_\theta)$  **involves an expectation according to the distribution  $p$**  which will be easily accessible via the data  $\{x_i\}_{i \leq n}$  representing *iid* samples from the distribution  $p$ :

$$I(p, p_\theta) \approx \frac{1}{n} \sum_{i=1}^n \left( \Delta_x U_\theta(x_i) + \frac{1}{2} \|\nabla_x U_\theta(x_i)\|^2 \right) \quad (138)$$

and we know the analytical formulas of  $U_\theta(x)$  so we can calculate the gradient and the

Laplacian without worry<sup>95</sup>. There is no longer a need for MCMC methods. In the case of a linear form of energy parametrization, we can obtain an analytical formula for  $I(p, p_\theta)$ . This method, which seems miraculous, often works, S. Mallat tells us, and let's recall that all synthesis/generation algorithms are based on this formulation.

**Proof 7.** The proof follows a classical process. If we write  $p(x) = Z^{-1}e^{-U(x)}$ , replacing  $\log p$  gives us  $-U(x)$  (the same for  $p_\theta$ ), thus

$$\begin{aligned} I(p, p_\theta) &= \frac{1}{2}\mathbb{E}_p[\|\nabla_x U(x) - \nabla_x U_\theta(x)\|^2] \\ &= \underbrace{\frac{1}{2}\mathbb{E}_p[\|\nabla_x U(x)\|^2]}_{C(p)} + \frac{1}{2}\mathbb{E}_p[\|\nabla_x U_\theta(x)\|^2] - \mathbb{E}_p[(\nabla_x U(x))^T \nabla_x U_\theta(x)] \end{aligned} \quad (139)$$

Now,

$$\begin{aligned} -\mathbb{E}_p[(\nabla_x U(x))^T \nabla_x U_\theta(x)] &= - \int p(x)(\nabla_x U_\theta(x))^T \nabla_x U(x) dx \\ &= - \int \nabla_x U_\theta(x)^T \underbrace{Z^{-1} \nabla_x U(x) e^{-U(x)}}_{\nabla_x p(x)} dx \\ &= \int (\nabla_x U_\theta(x))^T \nabla_x p(x) dx \\ &\stackrel{IPP}{=} \int p(x) \Delta_x U_\theta(x) dx \end{aligned} \quad (140)$$

Therefore,

$$I(p, p_\theta) = C(p) + \frac{1}{2}\mathbb{E}_p[\|\nabla_x U_\theta(x)\|^2] + \mathbb{E}_p[\Delta_x U_\theta(x)] \quad (141)$$

which is the expected result, if  $p(x)\nabla_x U_\theta(x)$  converges to 0 as  $\|x\| \rightarrow \infty$ , which is indeed verified thanks to the chosen regularity condition. ■

## 6.5 Example: Linear Energy Case

This result from Th. 7 is remarkable, let's recall it, as it eliminates all normalization constants. Let's see what this gives in the case of a linear expression of the energy in terms of the parameters. We know that minimizing the Kullback-Leibler divergence will

---

95. NDJE. Moreover, with automatic differentiation software.

work perfectly. In this case,  $U_\theta(x) = \theta^T \Phi(x)$ , so omitting the term  $C(p)$ , we can focus on the term dependent on  $\theta$ :

$$J(\theta) = \theta^T \mathbb{E}_p[\Delta_x \Phi(x)] + \frac{1}{2} \mathbb{E}_p[\|\theta^T \nabla_x \Phi(x)\|^2] \quad (142)$$

which is a convex quadratic expression in  $\theta$ , so there is no difficulty in minimizing it. The gradient of  $J$  is:

$$\nabla_\theta J(\theta) = \mathbb{E}_p[\Delta_x \Phi(x)] + \mathbb{E}_p[\nabla_x \Phi(x)(\nabla_x \Phi(x))^T] \theta \quad (143)$$

which gives the solution for  $\nabla_\theta J(\theta) = 0$ :

$$\theta_{SM} = -\mathbb{E}_p[\nabla_x \Phi(x)(\nabla_x \Phi(x))^T]^{-1} \mathbb{E}_p[\Delta_x \Phi(x)] \quad (144)$$

Thus, we have moved from a problem that could be long to compute at each step of the gradient descent to an analytical problem. However, if the feature vector  $\Phi$  is of high dimension, it is better to use gradient descent, using at step  $t$  the above expression of  $\nabla_\theta J(\theta)$  for  $\theta = \theta_t$ .

## 6.6 Convergence of Score Matching

First of all, we can ask the question: do we converge to a  $\theta$  identical to the one we would obtain by minimizing the Kullback-Leibler divergence? The brief answer to this question is "yes, if the model is correct...".

### Theorem 8 (Consistency of Score Matching)

*Imagine that  $p(x)$  is written as an element of the family  $\{p_\theta\}_\theta$  of the model in which we operate the search program for the best approximation. So let  $\theta^*$  be such that  $p = p_{\theta^*}$  with  $p_{\theta^*} > 0$  (e.g., exponential family), and let's assume that  $\theta^*$  is unique. Under these conditions,  $\theta_{SM} = \theta^*$  is reached when  $I(p, p_\theta) = 0$ .*

**Proof 8.** We know that by gradient descent, we reach the minimum of  $J(\theta)$ , which is

achieved when  $I(p_\theta, p_{\theta^*}) = 0$ . So for all  $x$ :

$$\nabla_x \log p_\theta(x) = \nabla_x \log p_{\theta^*}(x) \quad (145)$$

which by integration gives:

$$\log p_\theta(x) = \log p_{\theta^*}(x) + \text{const} \Rightarrow p_\theta(x) = \text{const} \times p_{\theta^*}(x) \quad (146)$$

Now, as both are pdfs, the constant is equal to 1. Thus, if the probability  $p(x)$  belongs to the family, we find the  $\theta$  corresponding to it by minimizing the Fisher divergence. ■

The problem at hand is not so much whether this method is more prone to the fact that  $p$  may not be an element of the probability family, **but do we indeed have a faster convergence rate, for example, than by minimizing the Kullback-Leibler divergence?**

S. Mallat draws a parallel with the seemingly miraculous theorem of universal function approximation by neural networks with 1-hidden layer<sup>96</sup>. Indeed, this theorem is ultimately completely ineffective because it is constrained by the curse of dimensionality: indeed, we can approximate any function, but the price to pay is that we need a number of parameters growing exponentially with the dimensionality of the problem. This is completely off-topic for  $d = 10^6$  in the case of image generation, for example.

So, the question of convergence speed is not trivial, and **in the case of score matching, we will see that this speed is governed by the number  $n$  of samples and therefore the precision we want to have for the calculation of the expectation.** That is, we make use of convergence  $((x_i)_i$  iid samples from  $p(x))$ :

$$\frac{1}{n} \sum_{i=1}^n F(x_i) \xrightarrow{n \rightarrow \infty} \mathbb{E}_p[F(x)] \quad (147)$$

for which we know that the fluctuations as a function of  $n$  are essentially Gaussian thanks to the central limit theorem.

If we return to the expression of  $\theta_{SM}$  in the linear case (Eq. 144), we can deduce,

---

96. NDJE. See Course 2019 Sec. 5.3

for example, that:

$$\sqrt{n}(\theta - \theta_{SM}) \xrightarrow{n \rightarrow \infty} \mathcal{N}(0, C_{SM}) \quad (148)$$

with  $C_{SM}$  a covariance that depends on the chosen metric, i.e., the Fisher divergence. Similarly, in the case of maximum likelihood estimation (MLE) using the minimization of the Kullback-Leibler divergence, thanks to the Cramér-Rao bound<sup>97</sup> of an unbiased estimator, we also have that:

$$\sqrt{n}(\theta - \theta_{KL}) \xrightarrow{n \rightarrow \infty} \mathcal{N}(0, C_{KL}) \quad (149)$$

with  $C_{KL} = 1/I_1(\theta)$  the inverse of the Fisher information obtained with 1 sample.

The question that arises is what is the relationship between  $C_{SM}$  and  $C_{KL}$ ? Let's first see an example before addressing the analysis of the problem itself.

## 6.7 An Illustrative Example of the Score Matching Problem

In fact, we try to control a log-probability by its gradient: indeed, to determine  $p_\theta$  or  $-\log p_\theta = Z_\theta + U_\theta$ , using the score matching method via  $I(p, p_\theta)$ , we eliminate the normalization constant and control the gradient  $\nabla_x U_\theta$ .

Let's imagine a function  $U(x)$  with local minima as shown in Figure 29, it's a non-convex problem. The question is whether we can perform a transformation of  $U$  without too much disturbing its gradient, while completely changing the minimum?

What we can do is to perform the transformation that gives the dashed red graph, which makes a cut of the potential barrier (hatched area) and a descent (by a constant) of the second minimum. There are areas where the gradients are little or not changed, and there is the cutting area that disturbs the gradients. But this last one is not the place of typical events which are rather concentrated in the minima. So, the evaluations of the expectations are not disturbed by the modified profile of  $U(x)$ . Or in other words, **to appreciate the differences between the profile of the initial  $U(x)$  or its modified form, a large number of samples is needed to explore the tails of distributions and explore the places where the gradients have been significantly modified.**

---

97. NDJE. See Course 2022 Sec. 5.3

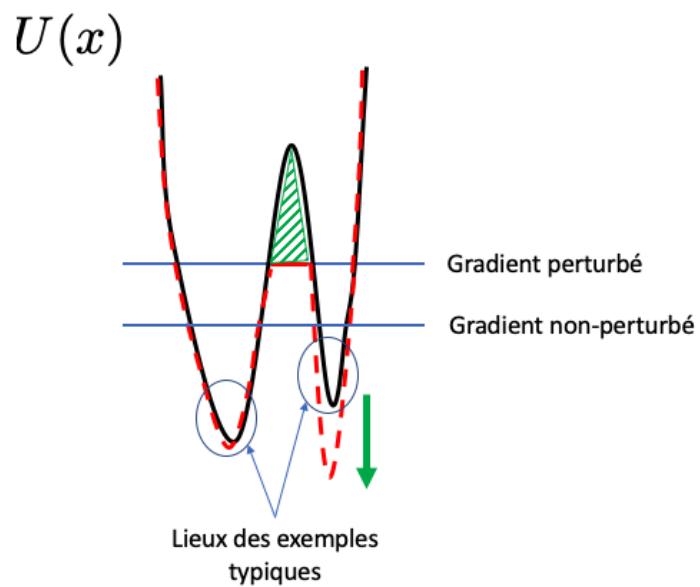


FIGURE 29 – Transformation of  $U(x)$  by a cut (hatching) and a translation of the secondary peak.

Thus, for non-convex problems, which is generally the case, there is a loss of knowledge of the exact profile of  $U(x)$  due to the non-use of the normalization constant. There is a risk of not converging to the correct value of  $\theta$ .

Another way to put it: the variance  $C_{SM}$  is likely to be very large. The question that arises then is: can we avoid this kind of situation?

## 6.8 Conditions for Using Score Matching

For the method to work, we need to control the Kullback-Leibler divergence by the Fisher divergence. It is in this framework that the notion of **log-Sobolev constant**<sup>98</sup> appears:

### Definition 5 (log-Sobolev)

Let  $p$  and  $q$  be two strictly positive probability densities, the log-Sobolev constant  $c(p)$  is the smallest constant such that:

$$\forall q, \quad D_{KL}(q||p) \leq c(p) I(p, q) \quad (150)$$

Note that we would rather have the idea of constraining  $D_{KL}(p||q)$  (recall:  $D_{KL}$  is not symmetric) which serves us when  $q = p_\theta$ . But we will see how this constant  $c(p)$  can be useful to us. Also note that the Fisher divergence is symmetric.

To see why we have this type of inequality, we will rewrite it like this (using the symmetry of  $I(p, q)$ ):

$$D_{KL}(q||p) = \int q(x) \log \frac{q(x)}{p(x)} dx \leq c(p) \frac{1}{2} \int \left\| \nabla_x \log \frac{q(x)}{p(x)} \right\|^2 q(x) dx \quad (151)$$

Let the function

$$f(x) = \sqrt{\frac{q(x)}{p(x)}} \quad (152)$$

---

98. NDJE. The log-Sobolev inequalities were discovered in 1967 by the American mathematician Leonard Gross (1931-) during his research into the construction of solid foundations for Quantum Field Theory, in particular the mutual interaction of bosonic fields involved in Quantum Chromodynamics, and more generally in the non-Abelian gauge theories of Yang-Mills. Publications appeared in 1975

we immediately have that  $\int f^2(x)p(x)dx = 1$ , and therefore  $f$  is an element of the set of square integrable functions, normalized with respect to the measure  $p(x)dx$  (denoted  $L^2(p(x)dx)$ ). Thus, we can rewrite the inequality as follows:

$$\int f^2(x) \log f^2(x) p(x)dx \leq \frac{c(p)}{2} \int \|\nabla_x \log f^2(x)\|^2 q(x)dx = 2c(p) \int \frac{\|\nabla_x f(x)\|^2}{\|f(x)\|^2} q(x)dx \quad (153)$$

Since  $q(x)/\|f(x)\|^2 = p(x)$ , the initial inequality then transforms into the following inequality<sup>99</sup>:

$$\int f^2(x) \log f^2(x) p(x)dx \leq 2c_{LS}(p) \int \|\nabla_x f(x)\|^2 p(x)dx \quad (154)$$

We are then immersed in another world of mathematics, that of functions whose square gradient is integrable according to the measure  $p(x)dx$ . The integral on the right-hand side is a Sobolev norm that ensures a form of regularity, while the left-hand side is an entropic measure. The challenge is then to control the constant.

Note that for practical applications of Machine Learning, **these inequalities are quite critical**, in the sense that **if we find ourselves in a situation that contradicts these inequalities, then score matching will not work at all!** So it is really important to understand the nature of these constants and to be aware of why we have come to use score matching/Fisher divergence instead of MLE/Kullback-Leibler divergence. In **Score Diffusion generation applications**, we use not an exponential model with linear energy in  $\theta$ , but a neural network whose output is  $s_\theta(x)$ . It is trained to compute  $\nabla_x \log p(x)$ , but **the question is how many samples do we need?** The answer to this question is largely influenced by these log-Sobolev constants.

To give an idea of these constants  $c(p)$ , let's link it with another classic inequality, namely the **Poincaré inequality**<sup>100</sup>, which relates the Sobolev norm to the variance.

---

99. NDJE. I denote the constant  $c_{LS}$  because later we will see another constant

100. Henri Poincaré (1854-1912)

**Theorem 9 (Poincaré inequality)**

Let  $g$  be a function, whose variance is considered under the measure  $p(x)dx$  and whose average is denoted by  $\bar{g}$ :

$$\text{Var}_p(g) = \int (g(x) - \bar{g})^2 p(x)dx \quad \bar{g} = \int g(x) p(x)dx \quad (155)$$

The Poincaré inequality states that the Sobolev constant (denoted here  $c_P(p)$ ) satisfies:

$$\text{Var}_p(g) \leq c_P(p) \int \|\nabla_x g(x)\|^2 p(x)dx \quad (156)$$

Typically, one thinks that the variance of  $g$  around its mean is related to the average of its variations, whose squared norm of the gradient is a proxy.

To link this inequality 154 with the one above, let's set  $f(x) = 1 + g(x).\varepsilon$  with  $\bar{g} = 0$ . Note that

$$f(x)^2 \log(f^2(x)) \approx 2(g(x).\varepsilon) + 3(g(x).\varepsilon)^2 + \dots$$

thus,

$$\int f^2(x) \log f^2(x) p(x)dx \approx 3\varepsilon^2 \text{Var}_p(g) \quad (157)$$

Similarly,  $\|\nabla_x f(x)\|^2 = \varepsilon^2 \|\nabla_x g(x)\|^2$ , so if we denote  $c_P(p)$  as the Poincaré constant and  $c_{LS}(p)$  as the log-Sobolev constant of expression 154, then we have a relation of the type<sup>101</sup>

$$c_P(p) = 2/3 c_{LS}(p) \quad (158)$$

The Poincaré inequality is very important in variational calculus<sup>102</sup>.

Let's consider applications. Take **the Gaussian case**, where  $g(x)$  is linear. Let  $e$  be a unit norm vector:

$$g(x) = \langle e, x \rangle = \sum_{i=1}^d e(i)x(i) \quad (159)$$

---

101. NDJE. It is found in the literature that  $c_{LS}(p)$  incorporates the factor  $1/2$  in the inequality Eq. 154, so the relation in this case would be  $c_P(p) = 1/3 c_{LS}(p)$ . In a more general case, keeping this definition, we find an inequality of the type  $c_P(p) \leq 1/2 c_{LS}(p)$ . See for example <https://djalil.chafai.net/blog/2023/01/12/log-sobolev-and-bakry-emery/>.

102. NDJE. For readers interested in the mathematical relations between the log-Sobolev inequalities also known as Gross and Poincaré inequalities, see <https://hal.science/hal-00012428v1>

the gradient of  $g$ ,  $\|\nabla_x g\|^2 = \|e\|^2 = 1$ . Now, let's look at the Poincaré inequality, which tells us

$$\forall e \text{ s.t. } \|e\| = 1, \quad \text{Var}_p(g) = \text{Var}_p(\langle e, x \rangle) \leq c_P(p) \quad (160)$$

We can choose the worst-case scenario where the unit vector  $e$  achieves maximum variance. What is the connection with the covariance matrix  $\text{Cov}(p)$  of  $p(x)$ ? In fact<sup>103</sup>,

$$\text{Var}_p(\langle e, x \rangle) = e^T \text{Cov}(p) e \quad (161)$$

Therefore, the eigenvector of  $\text{Cov}(p)$  with the largest eigenvalue (denoted  $\lambda_{max}$ ) achieves the worst-case scenario. Hence,

$$c_P(p) \geq \lambda_{max} \quad (162)$$

Returning to the problem of the  $\phi^4$  potential (Sec. 5.5) that is non-convex, at the moment of transition, long-range correlations appear. Since we are dealing with a stationary process, the matrix is diagonalizable in a Fourier basis. The eigenvalues constitute the spectral power<sup>104</sup>, which has a typical behavior of  $1/|\omega|^\eta$  (Fig. 30) with  $\eta \approx 2$  due to discontinuities<sup>105</sup> at small scales. So, at low frequencies, the covariance matrix is ill-conditioned. This reflects that in regions of constant gray levels (color), the pixels are highly correlated, so there are long-range correlations (large scale  $\leftrightarrow$  small value of  $\omega$ ), and a lot of power. **Thus, the constant  $c(p)$  is inherently large and therefore we will not effectively constrain the Kullback-Leibler divergence: "things go wrong!"**

However, how is it that neural networks manage to cope with this? Or how to circumvent this problem of poorly conditioned covariance matrix? In a way, the high frequencies (small scales) are masked by the low frequencies (large scales). *NDJE. In Course 2023 Sec. 9.3 we saw that one way is to consider windows in  $\omega$  on which the ratio of min/max eigenvalues is much more favorable (in fact, we then focused on finding a consistent estimator of second-order moments).* We can then perform **BatchNorm**<sup>106</sup> operations at all stages of the network, which will **(re)condition the problem to some extent to bring out the high frequencies.**

103. NDJE. It's a result that can be derived from Sec. 9.3 of Course 2023.

104. NDJE. Course 2023 Sec. 9.3

105. NDJE. There is a typo in the caption of figure 10. There is a behavior of  $1/k$  at small  $k$  i.e., at large spatial scales, and  $1/k^2$  at large  $k$  i.e., small scales.

106. NDJE. Cours 2019 Sec. 8.2.3

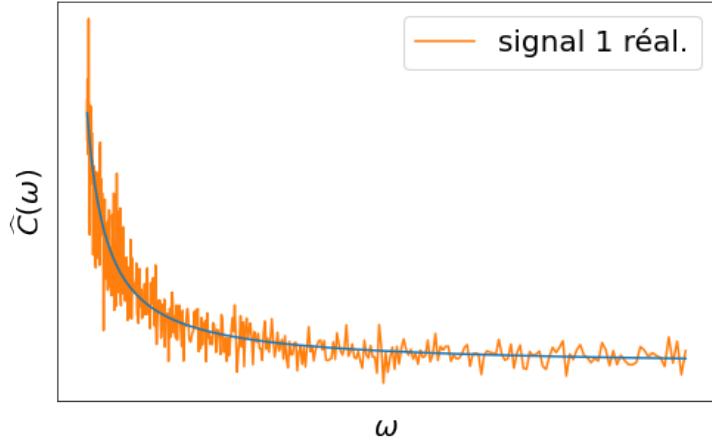


FIGURE 30 – Typical evolution of the spectral power, for example in images. (*NDJE. ex Course 2022 Figs. 10 and 11*)

There are two reverse inequalities to upper bound the Sobolev constants  $c(p)$ . The challenge is to be able to say in which cases these constants are not too large. There are 2 cases in which we can make these calculations. The first concerns **the convex case** of  $U(x)$  and we want to control the Hessian with respect to  $x$ . Let us then consider the theorem of Dominique Bakry and Michel Émery:

**Theorem 10 (Bakry-Emery, 1985)** *Let  $H_x[U(x)]$  be the Hessian of  $U$ . If  $\lambda_{min} > 0$  the smallest eigenvalue of this Hessian is such that*

$$\forall x, \quad H_x[U(x)] \geq \lambda_{min} \text{ Id} \quad (163)$$

then

$$c_{LS}(p) \leq 2/\lambda_{min} \quad (164)$$

This first result shows that in the Gaussian case ( $g$  linear), the log-Sobolev constant (Eq. 162) of Poincaré's theorem satisfies the equality

$$c_P(p) = \lambda_{max} \quad (\text{Gaussian case}) \quad (165)$$

(recall. where  $\lambda_{max}$  is the largest eigenvalue of the covariance) and the same is true<sup>107</sup> for  $c_{LS}(p)$  in the expression 154. In the non-Gaussian case,  $U(x)$  is not convex and the previous theorem does not apply.

The second case where we have a result is **the case of independence of components**  $(x(i))_{i \leq d}$  of  $x$ . Then we know that we can factorize  $p(x)$  according to

$$p(x) = \prod_{i=1}^d p_i(x(i)) \quad (166)$$

In this case, the Sobolev constant satisfies the following inequality:

$$c_{LS}(p) \leq \max_{i \leq d} c_{LS}(p_i) \quad (\text{independent variables case}) \quad (167)$$

How does this help to solve most of the problems? We have seen that problems arise when the dimension  $d$  increases. **In fact, this translates into an increase in these log-Sobolev constants with  $d$ . It's the curse of dimensionality.** But, in the case of independent variables, they do not interact, and therefore to escape local minima in  $d$  dimensions, it suffices to escape in only 1 dimension. *So, the phenomena of non-convexity that lead to more unstable constants become somewhat more manageable in 1D.*

But once again, in most high-dimensional problems, as  $d$  increases in a certain way, the minima become deeper and deeper, severely degrading the constants as stated for example by C. Domingo-Enrich and A. Pooladiani<sup>108</sup>: "[...], for multimodal distributions such as Gaussian mixtures,  $c_{LS}$  grows exponentially with the height of the potential barrier between the modes...".

We will come back to these considerations because **these log-Sobolev constants define the asymptotic speed of the Langevin sampling algorithm**. This algorithm performs gradient descent (along  $x$ , cf. score matching) while adding noise. So, when it is in a local minimum, it can "jump" the potential barrier by a fluctuation of the noise. But of course, the higher the barrier, the lower the transition probability. Then come the  $c_{LS}(p)$

107. NDJE. see Eq.1.4 <https://perso.math.univ-toulouse.fr/cattiaux/files/2013/11/CFG-21-06-22.pdf> with the change in definition of  $c_{LS}(p)$ .

108. NDJE. the excerpt is taken from the article by Carles Domingo-Enrich and Aram-Alexandre Pooladiani published in Transactions on Machine Learning Research (06/2023) "An Explicit Expansion of the Kullback-Leibler Divergence along its Fisher-Rao Gradient Flow", <https://arxiv.org/abs/2302.12229>.

to quantify these phenomena and control the convergence speed of the algorithm.

## 7. Session of Feb. 28th

### 7.1 Introduction

In this session and the following ones, we will address the issues of sampling and generation. The mathematical tools are of very different nature from those seen so far.

We assume that the determination of the underlying probability  $p(x)$  for the samples given initially is already acquired. To be concrete, we assume that  $p$  is a Gibbs distribution, namely

$$p(x) = Z^{-1} e^{-U(x)} \quad (168)$$

The problem now is to **generate realizations of this distribution**. Let these new<sup>109</sup> samples be  $\{x_i\}_{i \leq n}$ . If we compute averages from these samples, we should retrieve the values of the expectations obtained with  $p(x)$ . Let  $p_n(x)$  be the *empirical distribution* forged from these samples:

$$p_n(x) = \frac{1}{n} \sum_{i=1}^n \delta_{x_i} \quad (169)$$

then we expect that for any integrable function

$$\int f(x) p_n(x) dx = \frac{1}{n} \sum_{i=1}^n f(x_i) \xrightarrow{n \rightarrow \infty} \int f(x) p(x) dx = \mathbb{E}_{x \sim p}[f(x)] \quad (170)$$

**The interest in generating samples is to approximate integrals in high dimensions with empirical means.** So, the problem is to generate  $\{x_i\}_{i \leq n}$  such that they span the entire configuration space  $\chi$  while focusing on the space where the probability is large<sup>110</sup>. The central idea to achieve the generation program is to create **dynamic systems** where the iterates  $x(t)$  are the  $\{x_i\}_{i \leq n}$ , in a sense where the index  $i$  is a "time". The underlying mathematics involve the notion of **ergodicity**<sup>111</sup>.

109. NDJE. Note: for convenience, we use the same notation as that of the initial samples, which may have led to the determination of the expression of  $p(x)$ .

110. NDJE. In the case of the existence of multiple modes, the geometry of this space is more complex.

111. NDJE. see Course 2023 Sec. 6.2

One-dimensional case is somewhat simpler, as we can define **transformations from a uniform measure** on  $[0, 1]$  to obtain the target probability density. However, it is necessary to define the ability to sample a uniform measure on  $[0, 1]$ , which leads us to consider the design of a **dynamic system that generates randomness from determinism**, and satisfies the ergodicity property. Alongside the "dynamic system" approach, we will explore **the rejection algorithm** (or "rejection sampling", or "hit & miss", etc.). In fact, there are cases where the "dynamic system" approach yields a distribution  $q(x)$  different from the desired one; in such cases, the rejection algorithm allows the transformation from  $q(x)$  to  $p(x)$ . This is a simple yet fundamental idea that applies in both low and high dimensions. Finally, we will address **importance sampling**<sup>112</sup> which is one of the basic sampling algorithms, allowing, from a distribution  $q(x)$ , to recover the sought-after distribution  $p(x)$ .

Now, transitioning to **high dimensions** makes us fear facing the curse: bluntly applying the above techniques does not work. We need to find a way to ensure that  $q(x)$  is very well adapted. The appropriate dynamic system is that of **Markov chains**, and the **Metropolis-Hastings algorithm** is a rejection algorithm applied to them. These two ingredients form the basis of **Monte Carlo Markov Chain** (MCMC) algorithms.

To conclude, we will address **Score Diffusion/Denoising algorithms** to see how they fit into the previous framework. The idea is to once again define Markov chains. However, we are more interested in  $-\nabla_x U(x) = \nabla_x \log p(x)$ , which is nothing but the score (Sec. 5.4). We will then see the connection with denoising. Neural networks come into play to approximate  $\nabla_x \log p(x)$  via a function  $s_\theta(x)$ . If everything is under control, then we are able to generate samples from a Markov chain according to the correct probability  $p(x)$  starting from white noise.

## 7.2 The One-Dimensional Case

Let's assume we have a random variable  $U$  with a uniform density on  $[0, 1]$  (denoted  $\mathcal{U}(0, 1)$ ). We want to be able to sample  $p(x)$  with  $x \in \mathbb{R}$ . Let **the partition function**  $F(x)$  be defined such that

$$F(x) = \int_{-\infty}^x p(u)du \tag{171}$$

---

<sup>112</sup>. NDJE. see the footnote 23.

From there, we use the following method:

**Lemma 2 (Inverse Transformation Method)**

Let  $F$  be defined  $\mathbb{R} \longleftrightarrow [0, 1]$ , strictly monotonic, and let  $U \sim \mathcal{U}(0, 1)$ , then

$$X = F^{-1}(U) \quad (172)$$

is a random variable whose cumulative distribution function is  $F(x)$  and thus density  $p(x) = F'(x)$ .

**Proof 2.** Consider the cumulative distribution function of  $X$ :

$$\mathbb{P}(X = F^{-1}(U) \leq x) = \mathbb{P}(U \leq F(x)) = F(x) \quad (173)$$

indeed  $F(x) \in [0, 1]$  and for all  $a \in [0, 1]$ ,  $U$  being uniformly distributed  $\mathbb{P}(U \leq a) = \int_0^a 1 dx = a$ . Therefore, we have the lemma's result. The "strict" side of monotonicity can be relaxed, in which case  $F^{-1}(x)$  must be defined such that

$$F^{-1}(x) = \inf\{x \text{ such that } F(x) \leq u\} \quad (174)$$

■

Note that  $p(x)$  can be complex but obtaining approximations (e.g., splines) for  $F(x)$  can make the inversion of  $F(x)$  easy. The real difficulty lies in sampling  $\mathcal{U}(0, 1)$ . The central theorem is that of **Birkhoff**<sup>113</sup> and the notion of **ergodicity**.

### 7.3 Ergodicity of a Deterministic Transformation

**Definition 6 (Ergodic Transformation)**

Consider a probabilistic ensemble  $(\chi, \mathcal{B}, \mu)$  consisting of a set of states  $\chi$ , a measure  $\mu$ , and a set of measurable sets relative to  $\mu$  (denoted  $\mathcal{B}$ ) which is a  $\sigma$ -algebra<sup>a</sup>. We assume that  $\mu$  is a probability measure on  $\chi$ , specifically  $\mu(\chi) = 1$ . A transformation

---

113. NDJE. See Theorem 11 in Course 2023 Sec. 6.2

$T$  is ergodic if:

$$\bullet \quad \forall A \in \mathcal{B}, \quad \mu(T^{-1}(A)) = \mu(A) \quad (\text{invariant measure}) \quad (175)$$

$$\bullet \quad A \in \mathcal{B}, \text{ such that } T^{-1}(A) = A \Rightarrow \begin{cases} \text{either } \mu(A) = 0 \\ \text{or } \mu(A) = 1 \end{cases} \quad (176)$$

---

a. NDJE. e.g., the open sets of  $\mathbb{R}^d$

NDJE. To understand the first property, let's consider for example  $A \in \mathcal{B}$ , its measure with respect to  $\mu$  can be written as

$$\mu(A) = \int_{\chi} \mathbf{1}_A(x) d\mu(x)$$

Let  $T(x)$  be the transformation of  $x$  by the application  $T$ , we may wonder what is represented by:

$$\int_{\chi} \mathbf{1}_A(T(x)) d\mu(x)$$

This is the measure of the set  $B$  containing all the transformed elements of  $\chi$  that end up in  $A$  (Fig. 31), so  $B = T^{-1}(A)$  and the integral above can be written as

$$\int_{\chi} \mathbf{1}_A(T(x)) d\mu(x) = \int_{\chi} \mathbf{1}_B(x) d\mu(x) = \mu(B) = \mu(T^{-1}(A))$$

The notion of measure invariance by  $\mu$  is then identified as the stated property, namely  $\mu(A) = \mu(T^{-1}(A))$  for all  $A$ . In measure theory, we define the notion of the pullback of the measure by  $T$ , denoted  $T_*\mu$ , and we then write

$$\int_{\chi} \mathbf{1}_A(T(x)) d\mu(x) = \int_{\chi} \mathbf{1}_A dT_*\mu$$

The invariance is then written as  $T_*\mu = \mu$ .

The second property indicates that there is no set  $A$  other than the entire set  $\chi$  or reduced to isolated points, such that taking an element  $x$  and proceeding with the iterates  $x_n = T^n(x)$ , we would remain trapped in this set.

The central theorem is that of **Birkhoff's ergodicity**, which is stated as follows:

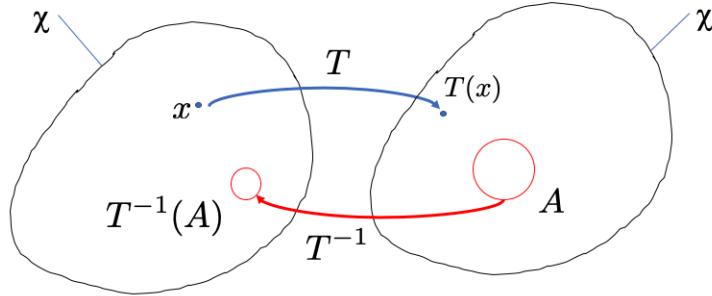


FIGURE 31 – Diagram showing why  $T^{-1}(A)$  appears in the definition of the invariant measure.

### Theorem 11 (Birkhoff)

If  $T$  is ergodic then for any measurable function  $f \in L^1(\chi, \mu)$ ,  $\int_{\chi} |f(x)| d\mu(x) < \infty$ , considering almost every  $x_0 \in \chi$

$$\frac{1}{n} \sum_{k=1}^n f(T^k(x_0)) \xrightarrow{n \rightarrow \infty} \int_{\chi} f(x) d\mu(x) = \mathbb{E}_{\mu}[f(X)] \quad (177)$$

The ergodicity assumption tells us that the empirical average calculated on the transformed points  $(x_k = T^k(x_0))_{k \leq n}$  converges to the expectation relative to the measure  $\mu$  of the random variable  $X$  generated by the  $(x_k)_k$ . In other words, the  $(x_k)_k$  will traverse the entire space and the concentration of these points in  $\chi$  reflects the probability density  $d\mu(x)$ .

An important special case is when  $f = \mathbf{1}_A$ , the theorem indicates (recall  $\mu(\chi) = 1$ ):

$$\frac{1}{n} \sum_{k=1}^n \mathbf{1}_A(T^k(x_0)) \xrightarrow{n \rightarrow \infty} \mathbb{E}_{\mu}[\mathbf{1}_A] = \mu(A) \quad (178)$$

which means that **the counting of returns in the set  $A$**  starting from  $x_0$  gives the measure of  $A$ .

**The program is as follows:** given a measure (e.g., the probability density  $p(x)$ ), find a transformation  $T$  that leaves this measure invariant, and such that the iterates traverse the entire space, meaning that its only invariant sets are either the set  $\chi$  or

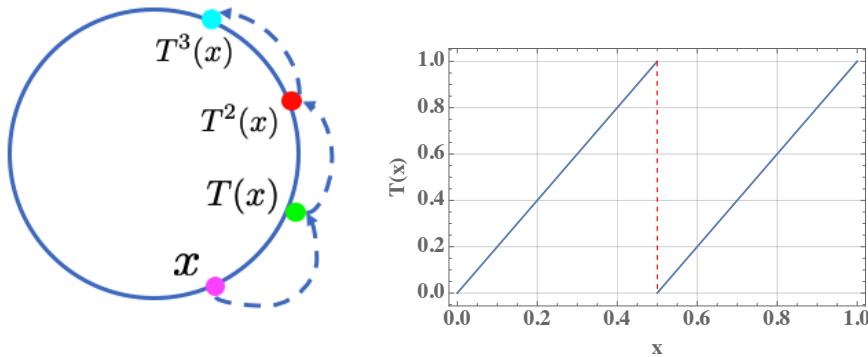


FIGURE 32 – Successive iterates by transformation  $T$  of a point  $x$  on the circle. On the right, the dyadic transformation Eq. 179.

**isolated points.** As mentioned in the introduction, if it is not possible to obtain  $p$  as the invariant measure of a transformation, one must find an invariant measure  $q$  close to  $p$ , which will be transformed by rejection-type algorithms.

## 7.4 Uniform Measure on $[0, 1]$

To start, we need to obtain samples from a uniform distribution on  $[0, 1]$ , denoted  $\mathcal{U}(0, 1)$ . For this, we can try to find a transformation  $T$  that maps a point from the unit circle to another point on this circle. We would like the density of points to be uniform on the circle.

Consider the following expression (Fig. 32):

$$T(x) = 2x \bmod 1 \quad (179)$$

which is a binary shift (*bit shift map*, or Bernoulli or *dyadic map*). If we start with an irrational  $x$ , in binary notation, the list of its bits is not periodic, so the shift produces a different number at each iteration, and therefore there is no periodicity. However, it is necessary to demonstrate that the invariant measure is the uniform measure<sup>114</sup>. Note that for two close irrational numbers, the iterates after a certain rank will diverge. This

---

114. NDJE. hints: If  $(b_0, b_1, b_2, \dots)$  is an infinite sequence of bits  $(0, 1)$ , then the transformation  $T$  yields

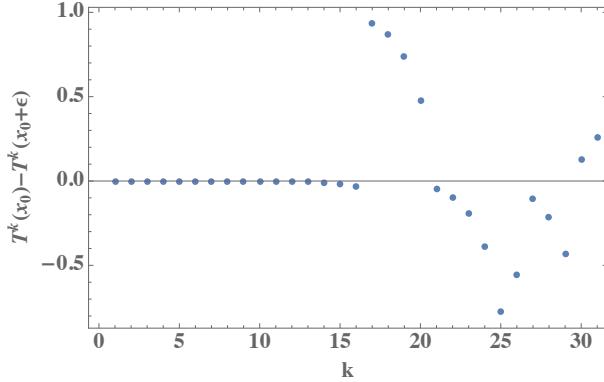


FIGURE 33 – Successive iterates by dyadic transformation  $T$  of  $x_0 = 1/\sqrt{2}$  and  $x_1 = x_0 + 10^{-6}$ .

is because in the decomposition of the two numbers, the first  $N$  bits are identical (see footnote 114), but then they differ. However, at the  $N$ -th iteration, only these differing bits matter, which leads to divergence. See an example in Figure 33. **This is a form of chaos that arises after a certain number of iterations of the dyadic transformation.**

Note that we can design an ergodic transformation with the uniform distribution as invariant by using the following expression for  $T$ :

$$T(x) = x - \tau \mod 1 \quad \tau \notin \mathbb{Q} \quad (180)$$

---

the sequence  $(b_1, b_2, \dots)$ , and any real number on  $[0, 1]$  can be written as

$$x = \sum_{k=0}^{\infty} \frac{b_k}{2^{k+1}}$$

An irrational number is characterized by a non-periodic sequence of bits. This demonstrates that  $T$  traverses the entire circle starting from an irrational number. Now, if we take a starting point density  $\rho(x)$ , one iteration transforms this density. To find the expression of the new density, we look for the points  $y = T^{-1}(x)$  (argument already discussed above). To obtain the density, we then use the derivative formula, taking into account the 2 intervals  $[0, 1/2]$  and  $[1/2, 1]$  involved by the modulo. We then find that the transformation of the density is written as

$$\rho(x) \xrightarrow{T} \frac{1}{2}\rho(x/2) + \frac{1}{2}\rho((x-1)/2)$$

Then note that  $\rho(x) = 1$  (uniform distribution) is invariant under this transformation. Now, if we only have a finite representation of irrational numbers, eventually the series of iterates will fall from a certain rank on the fixed point of the transformation, namely  $x = 0$ .

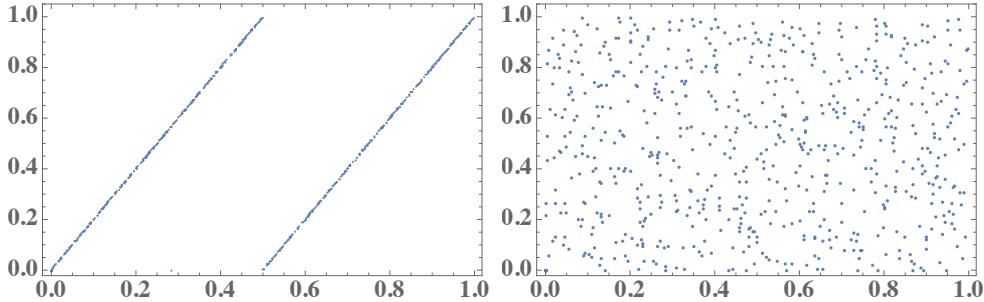


FIGURE 34 – Scatter plot of  $(X_n, X_{n+k})$  for  $k = 1$  (left) and  $k = 10$  (right) and  $X_0 = 1/\sqrt{2}$  and  $n \leq 1000$ . It can be observed that when  $k$  becomes sufficiently large, we approach a case where  $(X_n, X_{n+k})$  are independent.

It satisfies all ergodicity criteria, and the uniform measure is invariant. However, this transformation is not chaotic, which will have a drawback as we will see later.

In fact, in most cases, we want to generate samples  $(x_i)_i$  according to  $p$  that are **independent**. Note that for the Birkhoff theorem, this is not necessary. If we define the random variable denoted  $X_n$  from  $X_{n+1}$  according to

$$X_{n+1} = T(X_n) \quad (181)$$

it is clear that  $X_{n+1}$  depends on  $X_n$  (note that  $T$  is deterministic). But after  $k$  iterations

$$X_{n+k} = T^k(X_n) \quad (182)$$

can there be a mechanism leading to the quasi-independence of  $X_{n+k}$  with respect to  $X_n$ ? **This is where chaos allows for this "de-correlation."** An illustration of this quasi-decorrelation phenomenon is shown in Figure 34 for the dyadic transformation. To verify the independence of  $X_n$  and  $X_{n+k}$ , statistical tests can be performed, such as dividing the interval  $[0, 1]$  into sub-intervals of equal length and counting the number of  $x_n$  in each sub-interval. In principle, if everything is correct, a flat histogram should be obtained (within statistical fluctuations of Poisson). However, there are more sophisticated tests<sup>115</sup>.

---

<sup>115</sup>. NDJE. Some algorithms and tests are discussed here <https://www.omscs-notes.com/simulation/generating-uniform-random-numbers/>. An in-depth analysis of technological aspects is provided here [https://theses.hal.science/tel-00759976v1/file/These\\_MathildeSoucarros.pdf](https://theses.hal.science/tel-00759976v1/file/These_MathildeSoucarros.pdf).

The two transformations mentioned are not very good in practice<sup>116</sup>, here is a better one<sup>117</sup> that does not rely on bit manipulation:

$$T(x) = 4x(x - 1) \mod 1 \quad (183)$$

This is called the *logistic map*. The iterations of this transformation lead to chaos by frequency doubling, with the invariant measure being  $q(x) = \frac{1}{\pi\sqrt{x(x-1)}}$ . Although this is not the uniform distribution, we can use its cumulative distribution function  $F(x)$  for this purpose (reversing Lemma 2). We then obtain  $F(x) = 2 \arcsin(\sqrt{x})/\pi$ . Thus, we can define the sequence  $X_{n+1} = T(X_n)$  and then  $Y_{n+1} = F(X_n)$  to obtain a sequence of numbers uniformly distributed in  $[0, 1]$ , as shown in Figure 35.

Now, the problem that will concern us is the transition to high dimensions. But even for  $d = 2$ , we encounter a basic problem because the cumulative distribution function  $F$  is defined  $\mathbb{R}^2 \rightarrow \mathbb{R}$  and thus cannot be inverted. Therefore, we need to consider other techniques.

---

116. NDJE. In the 1990s, large experiments at CERN's LEP needed to significantly increase the number of simulations of their detectors (DELPHI, ALPEH, OPAL, L3). Moreover, to take advantage of multiple computing centers outside of CERN (e.g., the CCIN2P3 in France), it was necessary to find random number generators (PRNGs) that were resistant enough to statistical tests to make these simulations independent. It is in this context that Martin Luescher proposed RANLUX in 1993 (see <https://arxiv.org/pdf/hep-lat/9309020.pdf>), with a period of the order of  $10^{171}$ , allowing for the distribution of seeds to different centers and enabling the desired independent generations. Without this, neither the LEP nor later the LHC experiments could have considered studying rare events requiring the most accurate simulations possible. Furthermore, RANLUX is theoretically robust. Now, C++ or Python/Numpy use MT19937, a PRNG of the Mersenne Twister type developed by Makoto Matsumoto in 1997, which also has advantages, with a colossal period of  $2^{19937} - 1 \approx 4 \times 10^{6001}$ , but it has drawbacks, particularly concerning seed distribution for independent Monte Carlo draws. However, cryptography has provided new algorithms that are at the core of libraries for generating random numbers, such as JAX. See, for example: John K. Salmon, Mark A. Moraes, Ron O. Dror, and David E. Shaw. (2011). *Parallel random numbers: as easy as 1, 2, 3*. In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11). Association for Computing Machinery, New York, NY, USA, Article 16, 1–12. <https://doi.org/10.1145/2063384.2063405>, accessible here <http://www.thesalmons.org/john/random123/papers/random123sc11.pdf>. Therefore, the message is: before embarking on programs that require a large number of random draws, it is better to check which basic PRNG is used.

117. NDJE. I correct the typos made in the table.

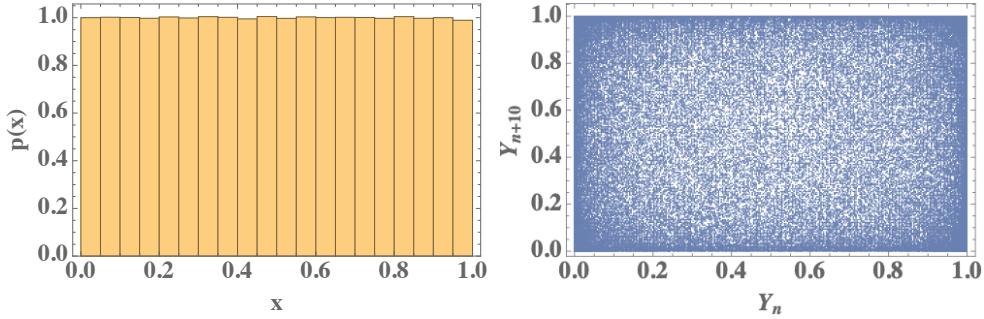


FIGURE 35 – Left: Histogram of  $Y_n$  generated by the logistic map sequence transformation (Eq. 183) using the cumulative distribution function of its invariant measure. We used  $X_0 = 0.1$  and generated  $10^6$  samples of  $Y_n$ . Right: The biplot  $(Y_n, Y_{n+10})$  for  $10^5$  samples. The distribution is quasi-uniform, and we can observe the effect of the fixed points  $x = 0$  and  $x = 1$  of  $T(x)$ .

## 7.5 Rejection Techniques

### 7.5.1 Simple Version

These are generic methods where the central idea is to enclose  $p(x)$  within a box. That is, for the variable  $x \in \text{support}(p(x)) \subset \mathbb{R}^d$ , we add a variable  $u \in [0, \max_x p(x)] \subset \mathbb{R}^+$ . The bounds on  $x$  and  $u$  form the dimensions of the box  $\mathcal{D}$ . Let  $\mathcal{A}$  be defined as follows:

$$\mathcal{A} = \{(x, u) / 0 \leq u \leq p(x)\} \subset \mathcal{D} \quad (184)$$

Consider the pair of random variables  $Z = (X, U)$  iid according to the uniform measure on the box  $\mathcal{D}$ . If  $p_Z$  denotes the probability density of  $Z$ , then the marginal distribution according to  $X$  is given by:

$$p_X(x) = \int_{\mathcal{A}} p_Z(x, u) du = \int_0^{p(x)} 1 du = p(x) \quad (185)$$

Thus,  $x$  follows the distribution  $p(x)$ . Therefore, the algorithm is as follows: The samples  $(x, u)$  are rejected if they are not in  $\mathcal{A}$ , i.e., if they belong to  $\mathcal{A}^c$ . They are accepted otherwise. See an illustration for  $d = 1$  in Figure 36.

This type of algorithm is extendable to cases where the "box"  $\mathcal{D}$  is replaced by a set

---

**Algorithm 1** Rejection sampling (1)

- 
- 1: Shoot uniformly  $(x, u) \in \mathcal{D}$
  - 2: **if**  $u > p(x)$  **then** reject
  - 3: **else** accept
- 

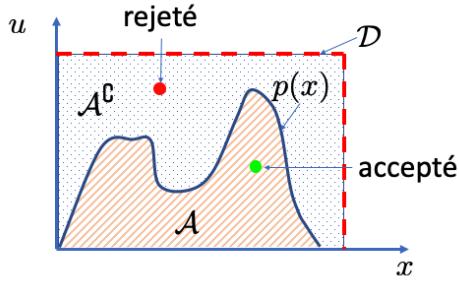


FIGURE 36 – Illustration of the rejection sampling method with  $p(x)$  in 1D.

that best fits the shape of the distribution  $p(x)$ .

### 7.5.2 Version based on $q(x)$

Let  $q(x)$  be a probability distribution from which we can sample, and let's find  $M$  such that:

$$p(x) \leq M q(x) \quad (186)$$

The rejection algorithm is then as follows (see Fig. 37 for a 1D version): The general

---

**Algorithm 2** Rejection sampling (2)

- 
- 1: **for**  $i : 1, \dots, n$  **do**
  - 2:   ok = False
  - 3:   **while** not ok **do**
  - 4:     Sample  $x_i \sim q(x)$  and  $u \sim \mathcal{U}(0, 1)$
  - 5:     **if**  $u \leq p(x_i)/(Mq(x_i))$  **then** ok = True
  - 6:     Keep  $x_i$
  - 7: Return  $(x_i)_{i \leq n}$
-

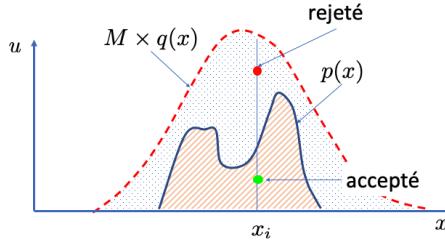


FIGURE 37 – Illustration of the rejection sampling method from a distribution  $q(x)$  for obtaining samples with  $p(x)$ .

principle is therefore the introduction of an auxiliary variable  $u$  independent of the draws  $x_i$ , which reduces the probability by the desired ratio to obtain a sample from  $p(x)$ <sup>118</sup>.

Now, the problem with this algorithm is that  $M$  should not be too large<sup>119</sup>. This is evident in the efficiency  $\eta$  of this algorithm defined using the ratio of the number of accepted samples  $x_i$  to the total number of attempts ( $n$ ):

$$\eta = 1 - \frac{\#\text{reject}}{\#\text{total}} = \frac{\#\text{accept}}{\#\text{total}} \quad (187)$$

### 7.5.3 Acceptance Probability

We need to compute<sup>120</sup>

$$\mathbb{E}_{x \sim q} \left[ \mathbb{P} \left( u \leq \frac{p(x)}{Mq(x)} | x \right) \right] = \mathbb{E}_{x \sim q} \left[ \frac{p(x)}{Mq(x)} \right] = \int \frac{p(x)}{Mq(x)} q(x) dx = \frac{1}{M} \quad (188)$$

118. NDJE. The marginal is written in this case as

$$f_X(x) = \int_0^1 \Theta \left( \frac{p(x)}{Mq(x)} - u \right) Mq(x) du = p(x)$$

with  $\Theta(x)$  the Heaviside function.

119. NDJE. See the case study discussed in the notebook [https://github.com/jecampagne/cours\\_mallat\\_cdf/blob/main/cours2023/Monte\\_Carlo\\_Sampling.ipynb](https://github.com/jecampagne/cours_mallat_cdf/blob/main/cours2023/Monte_Carlo_Sampling.ipynb) of a distribution  $p(x)$  with 2 modes, one of which is highly concentrated.

120. NDJE. For  $\alpha \in [0, 1]$ ,  $\mathbb{P}(u \leq \alpha) = \alpha$  for a uniform distribution on  $[0, 1]$ .

**So, if  $M$  has to be large to accommodate the value of  $\max_x p(x)$ , then the algorithm loses efficiency.** However, in high dimensions, we have phenomena of probability concentration (Sec. 3.2.4) which mechanically leads to large values of  $p(x)$  and hence  $M$ . **Typically, we expect**<sup>121</sup> **that**  $M \propto e^d$ . For example, by this argument, we can practically see that we cannot generate "faces" from a Gaussian distribution. In fact, we need to obtain a  $q(x)$  that is as close as possible to  $p(x)$ . Before seeing how to proceed, let's look at another type of sampling.

## 7.6 Importance Sampling

The idea is to adapt  $q(x)$  to compute an integral using Monte Carlo methods. Suppose we not only know how to compute  $p(x)$  and  $q(x)$ , but we also know how to sample from  $q(x)$  (note: we assume  $q(x)$  does not vanish on the support of  $p(x)$ ). Consider the following integral

$$I(f) = \int f(x) p(x) dx = \int f(x) \underbrace{\frac{p(x)}{q(x)}}_{w(x) \geq 0} q(x) dx = \mathbb{E}_q[f(x)w(x)] \quad (189)$$

Let  $(x_i)_{i \leq n}$  be independent samples drawn from  $q(x)$ , then

$$I_n(f) = \frac{1}{n} \sum_{i=1}^n f(x_i)w(x_i) \xrightarrow{n \rightarrow \infty} I(f) \quad (190)$$

$I_n(f)$  is an unbiased estimator due to the independence of the samples  $(x_i)_i$ , indeed we easily have

$$\mathbb{E}_q[I_n(f)] = \frac{1}{n} \times n \times \mathbb{E}_q[f(x)w(x)] = I(f) \quad (191)$$

The question then arises as to what is the convergence rate? We know that due to the independence of the random variables  $f(x_i)w(x_i)$  we have

$$Var(I_n(f)) = \frac{1}{n} Var(f(x)w(x)) \quad (192)$$

---

121. NDJE. An example is provided in the notebook mentioned in the footnote 39 in the case of evaluating the variance of the uniform measure on the unit ball in dimension  $d$ . We revisit Figure 9.

We would like the term  $\text{Var}(f(x)w(x))$  not to be too large. Let's try to bound it:

$$\text{Var}(f(x)w(x)) = \mathbb{E}_q[|f(x)w(x) - I(f)|^2] = \mathbb{E}_q[|f(x)w(x)|^2] - I(f)^2 \leq \mathbb{E}_q[|f(x)w(x)|^2] - I(f)^2 \quad (193)$$

equality being achieved if and only if  $|f(x)|w(x) = c$ . We would like  $q(x)$  to satisfy

$$|f(x)| \frac{p(x)}{q(x)} = c \Rightarrow q(x)c = |f(x)|p(x) \Rightarrow c = \int |f(x)|p(x) dx \quad (194)$$

thus the "best"  $q(x)$  is given by

$$q(x) = \frac{|f(x)|p(x)}{\int |f(x)|p(x) dx} \quad (195)$$

**This tells us that to compute  $I(f)$ , we can do better than sampling according to  $p(x)$ .** However, note that the initial calculation involves the integral of  $f(x)p(x)$ , and the normalization constant of  $q(x)$  is very close to it. So, while we have the expression for the best  $q(x)$ , in practice, we have a problem. We will then proceed with using a non-optimal  $q(x)$ .

**Note that in this type of method, we do not reject samples drawn from  $q(x)$ , but we directly calculate the empirical average in Eq. 190.** In fact, we calculate an average with a weighted empirical measure:

$$\tilde{p}_n = \frac{1}{n} \sum_{i=1}^n \delta_{x_i} w(x_i) \quad (196)$$

This method can be considered in the case where  $p(x) = Z_p^{-1}\tilde{p}(x)$  and  $q(x) = Z_q^{-1}\tilde{q}(x)$ , and where  $Z_p$  and  $Z_q$  are the normalization constants that are very difficult to estimate (e.g., Gibbs energy). Because then we just need to consider  $w(x) = \tilde{p}(x)/\tilde{q}(x)$  and

$$I_n(f) = \sum_i \frac{f(x_i)w(x_i)}{\sum_i w(x_i)} \quad (197)$$

(note: the same method can be applied if, of course, we know how to compute  $q(x)$  directly).

The limitation of this kind of method is similar to that encountered for the rejection

method, namely, to have an algorithm of good efficiency, we need to start from a quasi-optimal  $q(x)$ . And this becomes more and more critical in high dimensions due to the concentration of  $p(x)$  on its typical sets. **It is necessary to ensure that**  $q(x) \propto |f(x)|p(x)$  **with a proportionality constant that does not explode.**

To do this, **we need to be able to adapt**  $q(x)$  and we cannot do so by picking from functions of known typology *a priori*, which roughly amounts to taking a multivariate Gaussian.

## 7.7 Beyond "Rejection": Progression of Ideas

If we inspect the rejection method (Algorithm 2), at each iteration to obtain a new sample  $x_i$ , we do not make use of the knowledge gained from previous draws. In particular, for  $k < i$ , we have obtained samples representative of  $p(x)$ . Thus, **over time, we should be able to take into account the learning of the target distribution.**

Note that due to the concentration of  $p(x)$  (Fig. 12), most  $x$  have almost zero probability. But let's say we have obtained an  $x_i$  with a notably non-zero  $p(x_i)$ , we potentially have a good candidate, and we would then like to see what samples around this one yield. It is clear that the rejection algorithm loses this memory. Conversely, **from  $x_i$  we will attempt a small step**  $x_i \rightarrow x_i + \delta$ . **The problem** that immediately arises is that in the case where  $p(x)$  **has multiple modes**, we may end up only exploring the mode in which  $x_i$  lies, if  $\delta$  is too small compared to the separation between modes. Therefore, **we need to have a non-zero probability of taking a large step** (Fig. 38).

So, we need to establish a deterministic map  $T(x)$  that allows us to account for how to "move", and whose invariant measure is  $p(x)$ , knowing that we can start from a priori information, namely a rough estimate of  $p(x)$ . This is the idea of **Markov chains**: moving locally or with a large jump is formalized by the transition probability from  $x_n$  to  $x_{n+1}$ , i.e., **we model the conditional probabilities**  $p(x_{n+1}|x_n)$ . The model initially is probably "bad", so **we need to evolve these conditional probabilities**. How? through the **rejection algorithm**. This is the idea behind the **Metropolis-Hastings algorithm**<sup>122</sup> which uses the

---

122. Nicholas Metropolis (1915-99), Wilfred Keith Hastings (1930-2016), and there are other authors associated with this algorithm such as Arianna W. Rosenbluth (1927-2020), Marshall Rosenbluth (1927-2003), Augusta H. Teller (1909-2000), and Edward Teller (1908-2003), all associated with the Manhattan Project (Sec. 3.2).

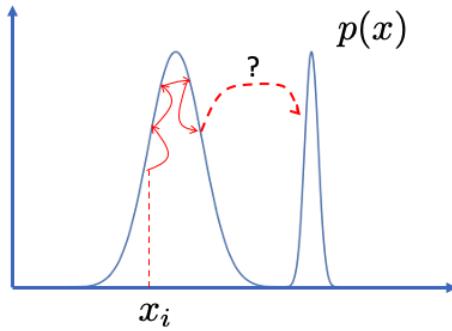


FIGURE 38 – Illustration of the method for exploring around a sample  $x_i$  with high probability. However, it is important to be able to take a large step to explore the other (or others) mode(s) of  $p(x)$ .

rejection algorithm on transition probabilities.

Now, if  $p(x)$  is a Gibbs probability, where  $p(x)$  is maximum  $U(x)$  is minimum and vice versa. So, having a sample  $x_n$ , it is clear that moving towards the minimum of  $U$  is guided by a **gradient descent!** Therefore, the transition probability would be wise to take into account  $-\nabla_x U(x)$ . But as in ML, we know that **adding noise** helps. However, here it is not so much about overcoming saddle points that motivates us, but rather the need to explore all possible modes of  $p(x)$  in order to be able to subsequently produce new samples<sup>123</sup>. This is the idea of the **Langevin algorithm**. However, if the amplitude of the noise is too large, we know that we cannot converge to the minima, and if it is too small, we are confined, so **adjustments need to be made**.

However, making adjustments to these limits, could we then **replace the multi-modal  $U(x)$  with a mono-modal  $U(x)$**  which therefore no longer has any issues with gradient descent. It sounds miraculous. And yet, that's the idea behind **Score Denoising**

123. NDJE.: There are different issues with gradient descent depending on its use. For a long time, understanding the results of neural networks has been a problem: when trying to imagine performing gradient descent in a high-dimensional universe, one might think that there are many local minima of the cost function to be minimized. Somehow, there is no guarantee that an SGD/Adam-type algorithm finds the global minimum. However, it is clear that differently trained networks, especially regarding the initialization seeds of the weights, have the same properties regarding the statistics performed on test/validation samples. There is a sort of universality of minima accessible by these algorithms that operate a regularization of the loss. So, exploring the entire space doesn't matter. The issue is completely different concerning obtaining samples from  $p(x)$  where all minima of  $U(x)$  need to be explored.

algorithms. We will gradually smooth  $U(x)$  to make it a convex function that can be easily sampled. So, we have a series of  $(U_i)_{i \leq n}$  with  $U_1 = U$  multi-modal and  $U_n$  mono-modal which are close to each other between two steps (underlying homotopy parameter). Then, conversely, we sample according to  $p_n = Z_n^{-1} e^{-U_n}$  and use the rejection algorithm to obtain samples from  $p_{n-1}$  and so on. The process of creating  $U_i$  is done by convolving with Gaussians. Now, making  $U_n$  as a quadratic form like  $x^T K x$  means having a Gaussian distribution for  $p_n$ , that is, noise! The samples from  $p_n$  are therefore noise. And so the process is first to gradually add noise, for example in a database of images, and then to proceed from a realization of pure noise to denoising to obtain a new image. **All this is done through a stochastic differential equation that we know how to invert.**

## 7.8 Markov Chain

### Definition 7 (Markov Chain)

A process  $(X_1, \dots, X_n)$  is a Markov chain if  $\forall (x_i)_{i \leq n+1} \in \chi^{n+1}$

$$\mathbb{P}(X_{n+1} = x_{n+1} | X_n = x_n, \dots, X_1 = x_1) = \mathbb{P}(X_{n+1} = x_{n+1} | X_n = x_n) \quad (198)$$

So, to produce  $X_{n+1}$  (future), we only need knowledge of  $X_n$  (present), or in more radical terms "the past doesn't matter". So, let's consider the following property/definition:

### Definition 8 (Stationary/Homogeneous)

A Markov chain is stationary (or homogeneous) if  $\forall (x, y) \in \chi^2$

$$\forall n, \quad \mathbb{P}(X_{n+1} = y | X_n = x) = \mathbb{P}(X_n = y | X_{n-1} = x) = P_{x,y} \quad (199)$$

so the transition probabilities do not depend on  $n$ , which can be denoted as  $P_{x,y}$  (careful with the order, think "going from  $x$  to  $y$ ")

How will this serve us? In particular, we hope to transition from a distribution  $p_0(x)$  to the target distribution  $p(x)$  we are looking for, which we will denote as  $\pi(x)$  afterwards, using the transition matrix to transform one or more samples from  $p_0(x)$  ( $x_0$ ), progressively

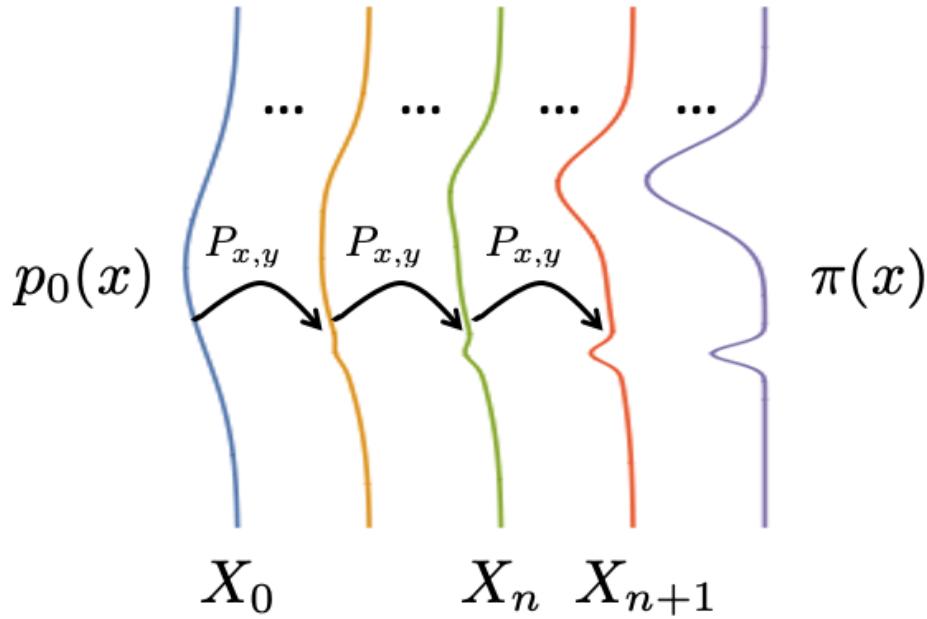


FIGURE 39 – Evolution of the initial distribution  $p_0(x)$  towards the target distribution  $\pi(x)$  by applying the transition matrix  $P_{x,y}$  of the Markov chain.

into  $x_1, x_2, \dots$ , that is, samples from the distributions  $p_1(x), p_2(x), \dots$  of random variables  $X_1, X_2, \dots$  (Fig. 39). **The whole problem is to determine the appropriate matrix  $P_{x,y}$  to achieve this goal.**

We have the following properties of the matrix  $P_{x,y}$ :

### Property 5

*In the case of an alphabet (i.e.,  $\chi$  is discrete), then  $(P_{x,y})_{(x,y) \in \chi^2}$  is a stochastic transition matrix with the following properties:*

— *Firstly*

$$\forall x \in \chi, \sum_{y \in \chi} P_{x,y} = 1 \quad (\text{stochastic matrix}) \quad (200)$$

*which comes from the fact that  $p(y|x) = p(x,y)/p(x)$  and  $\sum_y p(x,y) = p(x)$ .*

— Secondly, noting that<sup>a</sup>

$$\forall y \in \chi, \quad \mathbb{P}(X_{n+1} = y) = \sum_{x \in \chi} \mathbb{P}(X_n = x) P_{x,y} \quad (201)$$

we can use a matrix form by grouping the probabilities at step  $n$  into a column vector  $\mu_n$ , thus

$$\mu_n := (\mathbb{P}(X_n = x))_{x \in \chi} \implies \mu_{n+1} = P^T \mu_n \quad (202)$$

---

<sup>a</sup>. This is easily seen by expressing  $\mathbb{P}(X_{n+1} = y)$  as a marginal of the joint probability of  $(X_{n+1}, X_n)$ .

So, when we iterate, we have then, noting  $P = P_{x,y}$ ,

$$\mu_{n+k} = (P^T)^k \mu_n \quad (203)$$

and to determine if there is convergence, we are naturally invited to study the **eigenvalues of this matrix  $P$** , which will inform us about the possible existence of an **invariant measure** that we hope to be  $\pi(x)$ .

## 8. Session of March 6

Regarding the Markov chains addressed in the previous section, we assume that the step of data modeling has been completed, for example, according to a Gibbs distribution, which we denote in this Markovian framework as  $\pi(x)$ . The problem focuses on generating (sampling) new samples. Therefore, we only have initial information from  $\pi(x)$ , which is generally a high-dimensional distribution. **We start with a distribution  $q(x)$ , easy to sample from, and we construct a transformation that transports samples from  $q$  to samples from  $\pi$ .**

If we make a brief aside on Score Diffusion algorithms, the problem is different because we start a step earlier. Given data  $\{x_i\}_{i \leq n}$ , we want to both estimate the distribution  $\pi(x)$  and perform sampling. Nevertheless, Markov chains are underlying. We somewhat reverse the path, starting from  $\pi(x)$  and progressively adding noise to the samples

$\{x_i\}_i$  (Ornstein-Uhlenbeck stochastic equation) to obtain samples from  $q(x)$  consisting of Gaussian white noise (i.e., samples from  $q = \mathcal{N}(0, \Sigma)$ ). The set of samples obtained at intermediate stages forms Markov chains. Then, we reverse the process by denoising new realizations of pure noise from  $q(x)$ , and at the end of the process, we retrieve samples from  $\pi(x)$ . Estimation via the *score* is done using a deep neural network during the final step of the stochastic process inversion.

So, common to both approaches, we have *probability distribution transport*, *transition probability calculation*, and *the notion of time inversion*, i.e., *reversibility*. One difference between the two methods concerns the invariance of the transition probability in the first method and its evolution on the contrary in the Score diffusion method.

## 8.1 Markov Chains: Example

We continue Section 7.8 where we left off. A reminder:  $P_{x,y}^T$  is the transport matrix of the vector  $\mu_n$  of the set of probabilities  $(\mathbb{P}(X_n = x))_{x \in \chi}$  (see Fig. 39). The examples of Markov chains that will interest us are those obtained by adding independent *random variables*: these are **generalized random walks** where the notion of "hidden" variables comes into play.

### Theorem 12 (Random Walks)

Let  $\{Z_i\}_{i \geq 1}$  be iid and independent rvs from  $(X_j)_{j \geq 1}$ , where the state at step  $n$  depends on a recurrent function such that

$$X_{n+1} = f(X_n, Z_{n+1}) \quad (204)$$

This is typical of stochastic differential equations where  $Z_n$  is a kind of noise inducing random transitions. Then,  $(X_1, \dots, X_n)$  is a stationary Markov chain.

Note. The proof was given in the 2023 Course Sec. 6.4.2.2 Th. 13. For examples already studied in 2023, the notebook [https://github.com/jecampagne/cours\\_mallat\\_cdf/blob/main/cours2023/randomwalk.ipynb](https://github.com/jecampagne/cours_mallat_cdf/blob/main/cours2023/randomwalk.ipynb) describes the process  $X_{n+1} = \rho X_n + Z_{n+1}$  in 1D where  $Z_{n+1}$  is a Rademacher rv. The case  $\rho = 1$  yields the Brownian random walk. The notebook [https://github.com/jecampagne/cours\\_mallat\\_cdf/blob/main/cours2023/urne\\_](https://github.com/jecampagne/cours_mallat_cdf/blob/main/cours2023/urne_)

*Ehrenfest.ipynb* simulates a gas of  $N$  particles in a box with  $Z_{n+1}$  a rv taking values in  $\{-1, +1\}$  as in Rademacher, but this time  $\mathbb{P}(Z_{n+1} = -1 | X_n = x) = x/N$ . We discover the notion of irreversibility.

An example related to the Score Diffusion method is the **Ornstein-Uhlenbeck process** (around 1930), which simulates a particle subject to friction. The variable  $X_n$  is typically the velocity at a discretized time  $n$ , and we have<sup>124</sup>

$$\underbrace{X_{n+1} - X_n}_{\text{acceleration}} = \underbrace{-\alpha X_n}_{\text{friction}} + \underbrace{Z_{n+1}}_{\text{Brownian noise}} \quad \alpha \in [0, 1] \quad (205)$$

The solution can be written as

$$X_{n+1} = (1 - \alpha)^{n+1} X_0 + \sum_{i=1}^n (1 - \alpha)^i Z_{n+1-i} \quad (206)$$

As  $n$  tends to infinity, the term depending on the initial conditions  $X_0$  tends to 0, and if  $Z_{n+1} \sim \mathcal{N}(0, \sigma^2)$  then

$$X_n \sim \mathcal{N}(0, \sigma_n^2) \quad \sigma_n^2 = \frac{\sigma^2}{\alpha} (1 - (1 - \alpha)^n) \xrightarrow{n \rightarrow \infty} \frac{\sigma^2}{\alpha} \quad (207)$$

From this small example, we illustrate two fundamental points:

- If we are in good configurations, **we can forget the initial conditions** ( $X_0$ ) for sufficiently large  $n$ .
- And most importantly, **we converge to a fixed measure** (here the normal distribution defined by the parameters  $\sigma$  and  $\alpha$ ).

Therefore, in general, from the potentially complex distribution  $\pi(x)$  (multi-modal), we converge to a multivariate normal distribution, for example (Fig 40). What are the conditions for the Markov chain to converge?

---

124. Notebook [https://github.com/jecampagne/cours\\_mallat\\_cdf/blob/main/2024/Ornstein\\_Uhlenbeck.ipynb](https://github.com/jecampagne/cours_mallat_cdf/blob/main/2024/Ornstein_Uhlenbeck.ipynb) provides a 1D example with an additional mean term affecting the evolution of  $X_n$ .

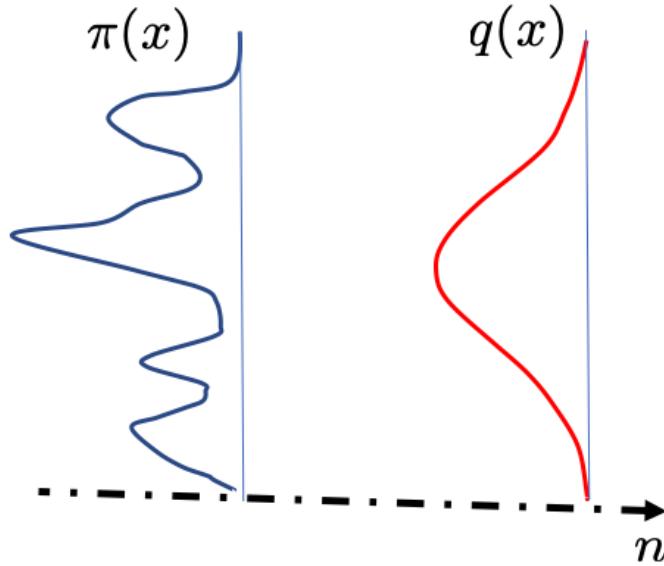


FIGURE 40 – Evolution from the initial distribution  $\pi(x)$  to a distribution  $q(x)$  (normal distribution) via the Ornstein-Uhlenbeck process.

## 8.2 Invariant Law

The convergence in law of the Markov chain relies on the existence conditions of a *stationary invariant* law.

### Definition 9 (*Stationary Invariant Law*)

*A law  $\pi$  is stationary invariant if it satisfies the following conditions, derived from the properties 5:*

$$\boxed{\pi = P^T \pi} \quad (208)$$

*This means that  $\pi$  is an eigenvector of  $P^T$  associated with the eigenvalue 1, and for each  $y$ , we have:*

$$\pi(y) = \sum_{x \in \chi} \pi(x) P_{x,y} \quad (209)$$

We thus seek to define  $P_{x,y}$  (and its transpose) so that  $\pi(x)$ , the given law, is indeed an eigenvector associated with the eigenvalue 1. Incidentally, we demonstrate the following

lemma:

**Lemma 3**

*If  $\pi(x)$  is invariant, and  $\mu_0 = \pi$ , then  $\forall n$ ,  $\mu_n = \pi$ .*

What we are particularly interested in is the concept of **time reversal**, which is the generation step in the Score Diffusion algorithm.

### 8.3 Reversibility Conditions

Recall that in the process of constructing the Markov chain, not only do the samples evolve over time ( $n$ ), but also their probability densities. Therefore, we transition from  $(X_0, \mu_0)$  to  $(X_1, \mu_1)$ , etc., which potentially converge to a random variable associated with the invariant measure  $\mu_\infty = \pi$ . Regarding the measures, the transport from  $\mu_0$  to  $\pi$  is deterministic (process related to  $P^T$ ), while concerning the  $X_i$ , it is a stochastic transport of random variables. Can we reverse these transformations?

We know that by definition of the matrix  $P_{x,y}$ :

$$P_{x,y} = \mathbb{P}(X_{n+1} = y | X_n = x) \quad (210)$$

We are interested in  $\mathbb{P}(X_n = y | X_{n+1} = x)$ . Using Bayes' theorem, we have:

$$\begin{aligned} \mathbb{P}(X_n = y | X_{n+1} = x) &= P(A|B) = \frac{P(B|A)P(A)}{P(B)} \\ &= \frac{\mathbb{P}(X_{n+1} = x | X_n = y)\mathbb{P}(X_n = y)}{\mathbb{P}(X_{n+1} = x)} \\ &= P_{y,x} \times \frac{\mathbb{P}(X_n = y)}{\mathbb{P}(X_{n+1} = x)} = Q_{x,y} \end{aligned} \quad (211)$$

If we have the good fortune to start the process with the invariant measure  $\pi$ , according to Lemma ??, then

$$Q_{x,y} = \mathbb{P}(X_n = y | X_{n+1} = x) = P_{y,x} \times \frac{\pi(y)}{\pi(x)}$$

(212)

**Definition 10** *Inverse Matrix*

For a stationary process (Definition 5) with a transition matrix  $P_{x,y} = \mathbb{P}(X_{n+1} = y | X_n = x)$  ( $\forall (x,y) \in \chi^2$ ). If there exists a stationary distribution  $\pi(x) \neq 0$ , meaning all  $x$  are accessible, then we define the following matrix  $Q_{x,y}$

$$Q_{x,y} = P_{y,x} \times \frac{\pi(y)}{\pi(x)} \quad (213)$$

The matrix  $Q$  is stochastic,  $\sum_y Q_{x,y} = 1$ .

(The property arises from  $\pi = P^T \pi$  and thus  $\pi(x) = \sum_y \pi(y) P_{y,x}$ ).

A particular but very practical case is that of reversible Markov chains:

**Definition 11** *Reversible Markov Chain*

We say that a stationary Markov chain with transition matrix  $P$  is reversible relative to an invariant distribution  $\pi$ , if under the conditions of Definition 10 then  $Q = P$ , which translates to

$$\pi(x)P_{x,y} = \pi(y)P_{y,x} \quad (\text{detailed balance}) \quad (214)$$

(where the global balance is given by the equation of the invariant measure  $\pi = P^T \pi$ ).

We then have the following diagram:

$$X_1 = x \sim \pi \xrightleftharpoons[P_{y,x}]{P_{x,y}} X_2 = y \sim \pi$$

Random walks as well as the Ehrenfest urn are examples of reversible processes. This notion of equality of flows from  $x \rightarrow y$  and from  $y \rightarrow x$  concerns the case of equilibrium physics (statistics)<sup>125</sup>. It leads to a property of the existence of an invariant measure:

---

<sup>125.</sup> NDJE. note that the march towards equilibrium is irreversible in high dimensions, see Course 2023.

**Property 6 (Existence of an invariant measure)** *If  $P$  is reversible (detailed balance) with respect to the measure  $\pi$ , then  $\pi$  is invariant.*

The proof is immediate: we know that  $\forall x, y, \pi(x)P_{x,y} = \pi(y)P_{y,x}$ , summing over  $y$  on both sides yields

$$\sum_y \pi(y)P_{y,x} = \sum_y \pi(x)P_{x,y} = \pi(x) \underbrace{\sum_y P_{x,y}}_{=1 \text{ (stochastic matrix)}} = \pi(x) \quad (215)$$

which is nothing but  $\pi = P^T\pi$ , thus  $\pi$  is indeed the invariant distribution.

So, concerning the objective of creating a Markov chain with the invariant measure being the distribution of interest, it will be to **construct transition matrices  $P_{x,y}$  that satisfy the detailed balance condition**. This is the objective of the Metropolis-Hastings algorithm.

Now, convergence properties are related to the notion of **ergodicity** (Definition ??) and to the **Birkhoff's theorem** (Theorem 11). Indeed, with the matrix  $P$ , we propagate a seed  $X_0$ , and the question is **whether we explore the entire space to explore the places where the probability  $\pi(x)$  is high**. In this case,  $X_0$  is not important. In short, we should not get "stuck" in regions of the  $X$  space (e.g., local energy minima in the case of Gibbs probabilities). We can translate the properties of stationarity and ergodicity of  $\pi$ : for any measurable space  $A$  (relative to  $\chi$  endowed with the measure  $\pi$ )

- $\pi(T^{-1}A) = \pi(A)$  (the measure is stationary/invariant)
- $T^{-1}A = A$ , then either  $\pi(A) = 0$  ( $A$  is reduced to singular points) or  $\pi(A) = 1$  ( $A$  is the entire set).

## 8.4 Convergence towards the invariant distribution

We will revisit Birkoff's theorem (Theorem 11) in the case of Markov chains. Firstly, it is necessary to ensure that we explore the entire set  $\chi$ :

**Definition 12 (Irreducible chain)**

A Markov chain is irreducible if one can transition from any state  $x$  to any other state  $y$  (and vice versa), i.e.,

$$\forall (x, y) \in \chi^2, P_{x,y} > 0 \quad (216)$$

The second property concerns the first return time:

**Definition 13 (First return time)**

For a state  $X_0 = x$ , we define the "time"  $T_x$

$$T_x = \inf\{n; X_n = x\} \quad (217)$$

( $T_x = +\infty$  if  $x$  cannot be reached)

As  $X_n$  is a random variable, the same applies to  $T_x$ , so we have the following definition:

**Definition 14 (Positive recurrence)**

A state  $x$  is said to be positively recurrent if the expectation of its return time is finite.

$$\mathbb{E}_x[T_x] < \infty \quad (218)$$

By extension, if all states are recurrent, then the chain is recurrent.

If this is the case, let's imagine that we are in an infinite set. The chain starting from  $X_0 = x$  will always return to this point in a finite time. Moreover, regarding the invariant measure  $\pi$ , we have for every  $x \in \chi$ :

$$\pi(x) = \frac{1}{\mathbb{E}_x[T_x]} \quad (219)$$

The last notion we need concerns the aperiodicity of the Markov chain. Consider the scheme in Figure 41: if we start from state "1" at  $n = 0$ , we return to it at  $n = 3, 6, \dots$ , and  $P_{1,1}^n = 1$  for  $n = 0 \pmod{3}$ .

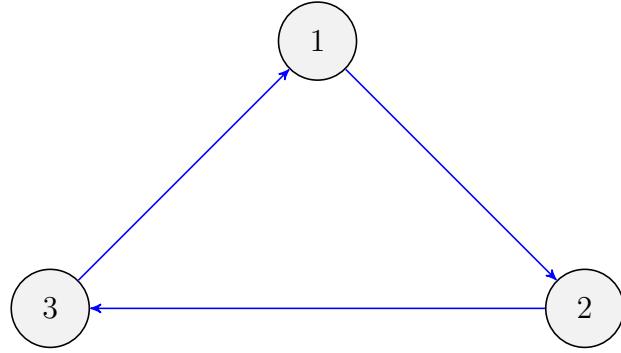


FIGURE 41 – Markov process with transition matrices having positive transitions between states.

Let's then define the period of a state:

**Definition 15 (period of a state, of a chain)**

Let  $\tau_x$  be the set of return times to a state  $x$

$$\tau_x = \{n \geq 1 \mid P_{x,x}^n > 0\} \quad (220)$$

The period of a state  $x$  is the greatest common divisor of  $\tau(x)$ . If the period is equal to 1, then it is said to be aperiodic.

By extension, a Markov chain is said to be aperiodic if all states  $x \in \chi$  are aperiodic.

This notion is important because it ensures that we can forget the initial conditions (i.e., the first state  $X_0$  of the chain). A sufficient condition is that

$$\forall x, \quad P_{x,x} > 0 \quad (221)$$

because then in 1 step we can return to the initial state. In the scheme in Figure 41, we would need to add arrows that loop from a state to itself (e.g., with a weight of  $1/2$ ).

So, to summarize, the considerations lead us to irreducible, recurrent, and aperiodic chains. We can then state without proof the fundamental theorem of Markov

chains<sup>126</sup>:

**Theorem 13 (Ergodicity)**

Let  $(X_n)_n$  be a Markov chain, irreducible and positive recurrent, then

1. the measure defined by

$$\pi(x) = \frac{1}{\mathbb{E}_x[T_x]} \quad (222)$$

is the unique invariant measure of the chain.

2. for any function  $f$ , such that  $\sum_x |f(x)|\pi(x) < \infty$ , then the Birkhoff theorem tells us that

$$\frac{1}{n} \sum_{k=1}^n f(X_k) \xrightarrow[n \rightarrow \infty]{p.s.} \sum_x f(x)\Pi(x) = \mathbb{E}_\pi[f(x)] \quad (223)$$

3. If furthermore the chain is aperiodic then

$$X_n \xrightarrow[n \rightarrow \infty]{law} \pi \quad (224)$$

that is

$$\mathbb{P}(X_n = x) \xrightarrow[n \rightarrow \infty]{} \pi(x) \quad (225)$$

The property 2 is practically useful for computing Monte Carlo integrals using the iterated samples of the Markov chain, while property 3 allows us to forget the initial condition of the chain<sup>127</sup>.

Therefore, the problem is as follows: if we are given  $\pi(x)$  (estimated in a modeling phase), we need to find  $P_{x,y}$  such that  $\pi$  is the invariant measure (detailed balance) and which will generate a Markov chain that must be irreducible, recurrent, and positive to be able to use the properties of the ergodicity theorem.

## 8.5 Spectral analysis viewpoint of the invariant measure

We have seen that the invariant measure is the eigenvector of  $P^T$  with an eigenvalue equal to 1 (see Def. 9). However, the drawback of Theorem 13 is that it does not provide

126. Note some minor differences compared to the 2023 version. Also, there is a typo in the 2023 version point 3) which I correct here, and the notes are updated on my GitHub page.

127. Note that in Course 2022 Sec. 5.5, there is a discussion about the terminology of convergences.

information about the convergence rate.

Consider then the classical analysis theorem

**Theorem 14 (Perron-Frobenius)**

Let  $P_{x,y}$  be a stochastic matrix (Eq. 200) with all entries strictly positive (irreducible chain Def. 12) having an invariant measure, then every eigenvalue  $\lambda$  of  $P_{x,y}$  satisfies  $|\lambda| \leq 1$ , and the eigenvalue 1 is simple.

Consequently, there exists a basis of eigenvectors  $(\pi_k)_k$  whose eigenvalues denoted  $(\lambda_k)_k$  ( $P^T \pi_k = \lambda_k \pi_k$ ) are such that  $|\lambda_k| \leq 1$ . Thus, the initial measure  $\mu_0$  (followed by the random variable  $X_0$ ) can be decomposed onto this basis as follows

$$\mu_0 = \sum_k \alpha_k \pi_k \quad (226)$$

Through successive transport, we have

$$\mu_n = (P^T)^n \mu_0 = \sum_k (\alpha_k \lambda_k^n) \pi_k \quad (227)$$

As  $n$  tends to infinity, all components such that  $\lambda_k < 1$  disappear, and only the component associated with the invariant measure remains, let's say  $\pi_0$ . How fast does the sum converge? **It is dominated by the eigenvalue whose modulus is the largest less than 1.** The *spectral gap* is defined as the difference  $1 - \max_{k \neq 0} |\lambda_k|$ , it governs the exponential decay of components different from the invariant measure.

The problem arises when dealing with high dimensions, controlling the spectral gap becomes difficult, and **we very often encounter a situation where eigenvalues accumulate towards 1 (in modulus) and exponential convergence becomes compromised.**

## 8.6 Metropolis-Hastings Algorithm

The method<sup>128</sup> first developed in 1949 by Nicholas C. Metropolis (1915-99) and Stanisław Ulam (1909-84) was further detailed in 1953 by Metropolis and collaborators, then

---

128. To introduce elements for using a notebook, I introduced this algorithm in 2023.

extended in 1970 by Wilfred Hastings (1930-2016). It is called the Metropolis-Hastings (MH) algorithm even though several authors contributed to it. It is an application of all the notions described in the previous sections.

We want the invariant probability density to be  $\pi(x)$  (this is the given problem). Recall: we can compute  $\pi(x)$ , and we will see that we can do without knowing how to compute the normalization constant, which is very appreciable in practice. If we manage to design a **reversible matrix**  $P_{x,y}$  (Def. 11), satisfying the equation of **detailed balance**, then  $\pi$  is the invariant measure, and with a bit of cleverness, we can meet the conditions of the **ergodicity theorem** (Th. 13).

To determine  $P_{x,y}$ , we make a **proposal** denoted  $Q_{x,y}$  which will be the transition matrix<sup>129</sup> from  $x$  to  $y$ ; we will choose it to be irreducible (recall: from any  $x$ , we can reach any  $y$ ) and strictly positive: for example,

$$Q(x, y) = (2\pi\sigma^2)^{-d/2} e^{-\frac{1}{2}\frac{\|x-y\|^2}{\sigma^2}} \quad (228)$$

Now, how to construct  $P_{x,y}$ ? The idea is to write

$$\boxed{P_{x,y} = \rho(x, y) \times Q(x, y)} \quad (229)$$

where  $\rho(x, y)$  is a function that we will adapt. It is a **rejection procedure** (Sec. 7.5) on the proposed transition probability. In this case,  $\rho(x, y) \in [0, 1]$ .

The objective of detailed balance is literally translated as

$$\rho(x, y)Q(x, y)\pi(x) = \rho(y, x)Q(y, x)\pi(y) \quad (230)$$

So, the following definition of  $\rho(x, y)$  is given

$$\boxed{\rho(x, y) = \min \left( 1, \frac{Q(y, x)\pi(y)}{Q(x, y)\pi(x)} \right)} \quad (\text{Metropolis-Hastings}) \quad (231)$$

---

129. Sometimes denoted  $Q(y|x)$ , and there are other notations. Pay attention to the order, check how  $P_{x,y}$  is decomposed according to the product of the simple distribution  $Q$  and the acceptance/rejection function.

Detailed balance is respected, and  $\pi$  is indeed the invariant measure.

Indeed, suppose that  $Q(y, x)\pi(y) > Q(x, y)\pi(x)$ , then  $\rho(x, y) = 1$ , and

$$\rho(y, x) = \frac{Q(x, y)\pi(x)}{Q(y, x)\pi(y)}$$

thus

$$P_{x,y} = Q(x, y) \quad P_{y,x} = Q(y, x) \times \frac{Q(x, y)\pi(x)}{Q(y, x)\pi(y)} = \frac{Q(x, y)\pi(x)}{\pi(y)} = \frac{P_{x,y}\pi(x)}{\pi(y)}$$

which clearly demonstrates that detailed balance is satisfied, and thus  $\pi$  is the invariant measure of the Markov chain associated with the transition matrix  $P_{x,y}$ . The symmetric case of the assumption is treated in the same way.

Note as announced that the calculation of  $\rho(x, y)$  is a ratio of probabilities where normalization constants are absent. This is very important in practice because, as we have seen, estimating the normalizations of Gibbs distributions involves integrals in high dimensions.

The algorithm unfolds as follows:

---

**Algorithm 3** Metropolis-Hastings

---

**Require:**  $Q(x, y)$  a distribution easy to sample to obtain  $x$

- 1: Shoot  $x_0 \sim \mu_0$  (e.g.,  $Q(x, 0)$ )
  - 2: **for**  $i : 1, \dots, n$  **do**
  - 3:     Shoot  $x_{prop} \sim Q(., x_{i-1})$  and  $u \sim \mathcal{U}(0, 1)$
  - 4:     Compute  $r = \rho(x_{i-1}, x_{prop}) = \min\left(1, \frac{Q(x_{prop}, x_{i-1})\pi(x_{prop})}{Q(x_{i-1}, x_{prop})\pi(x_{i-1})}\right)$
  - 5:     **if**  $r = 1$  OR  $u \leq r$  **then**  $x_i = x_{prop}$
  - 6:     **else**  $x_i = x_{i-1}$
  - 7:     Keep  $x_i$
  - 8: **return**  $(x_i)_{i \leq n}$
- 

This algorithm defines the following transition matrix  $P_{x,y}$ :

$$P_{x,y} = \rho(x, y)Q(x, y) + \alpha(x)\mathbf{1}(y = x) \quad \alpha(x) = 1 - \sum_{z \in \chi} \rho(x, z)Q(x, z) \quad (232)$$

(it is easy to verify that  $\sum_y P_{x,y} = 1$ ).

Now, the practical question is: **does the algorithm converge?** We know that the invariant measure is correct by construction because **detailed balance is ensured**, as we have demonstrated above. So, the question is whether the assumptions of the ergodicity theorem are satisfied?

We limit ourselves to states  $x \in \chi$  where  $\pi(x) > 0$  (those with zero probability are not to be considered). Now,  $Q(x, y) > 0$  (by choice), so  $\rho(x, y) > 0$ , hence  $P_{x,y} > 0$  for any pair of states, making the chain **irreducible**. Regarding aperiodicity, for  $P_{x,x} = 0$  to hold,  $\alpha(x)$  would need to be 0, implying that  $\sum_z \rho(x, z)Q(x, z) = 1$ , i.e.,  $\rho(x, z) = 1$  (always accepting), which would make  $Q$  the desired probability distribution. Thus,  $P_{x,x} > 0$ , which is a sufficient condition for the **aperiodicity** of the chain (note that  $P_{x,x} > 0$  is the algorithmic consequence that during the process, there are cases where the proposal is rejected, and thus  $x_{i+1} = x_i$ ). Since  $\pi(x) > 0$ , we deduce that  $\mathbb{E}_\pi[T_x] < \infty$ , so the chain is **positively recurrent**. Thus, **the conditions of the ergodicity theorem are met, and we indeed have convergence to the desired distribution.**

What about the convergence speed<sup>130</sup>? Asking this question is not trivial, as it is where much of the practice is based on experimentation<sup>131</sup>. The crucial point is that **convergence can be very slow**. Especially due to the weakness of the *spectral gap* in high dimensions, which can completely hinder convergence in a finite time. In fact, it is essential to judiciously choose  $Q(x, y)$ .

The first thing we can immediately see that will help us understand the problem is the particular case where  $Q(x, y)$  is symmetric (original Metropolis algorithm). In this case, we have

$$\boxed{\rho(x, y) = \min \left( 1, \frac{\pi(y)}{\pi(x)} \right)} \quad (\text{Metropolis}) \quad (233)$$

Now, as already mentioned, in the absence of *a priori* information about  $\pi(x)$ , one generally chooses  $q(x) = Q(., y)$  as a multivariate Gaussian with mean  $y$ , so the roles of  $x$

---

130. For those who want to delve deeper, see for example the article by Guanyang Wang from 2021 "*Exact Convergence Rate Analysis of the Independent Metropolis-Hastings Algorithms*" <https://arxiv.org/pdf/2008.02455v6.pdf>.

131. In particular, throughout practice, concepts such as *burning* emerged to consider only samples from a certain rank onwards, and *thinning* algorithms operated after this *burning* phase. The simplest thinning algorithm considers retaining every  $k$ -th iteration and discarding the rest. The goal is to reduce the positive correlation between the remaining states, thereby reducing the asymptotic variance of the estimators. However, this practice can be very delicate.

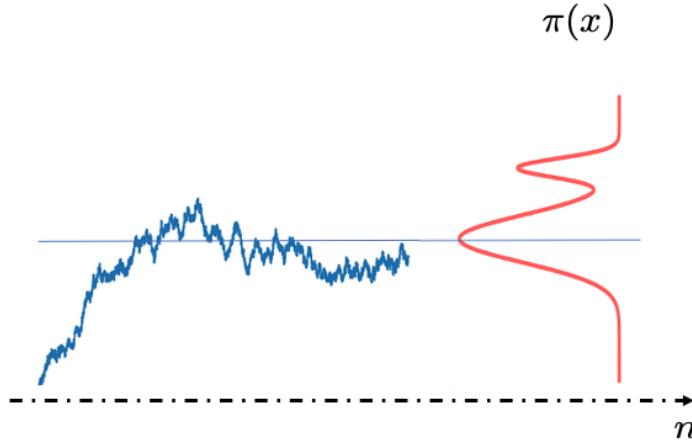


FIGURE 42 – Evolution of a chain produced by the Metropolis algorithm with samples only exploring one mode of the distribution  $\pi(x)$ .

and  $y$  are interchangeable. But, if  $\pi(x)$  has two modes (Fig. 42)<sup>132</sup>. The chain production begins, and suppose the sigma of the Gaussian is small, if we end up accumulating samples with a high probability due to the presence of one mode, then the ratio  $\pi(y)/\pi(x)$  will be close to 1, and we will always accept samples that only sample this mode. In other words, we have almost no chance of visiting the second mode.

To remedy this, one way or another, we need to ensure that the support of  $\pi(x)$  is sampled: for example, we can use a distribution for  $\mu_0$  with a larger width than  $Q(x, y)$  and evolve multiple chains ( $k \leq K$ ) in parallel from initial seeds  $x_0^{(k)} \sim \mu_0(x)$ . This is the approach taken to produce Figure 43.

However, the main problem arises when in high dimensions the support of  $\pi$  is concentrated in a small fraction of the space, and many chains may spend time in areas where  $r \leq 1$ , mechanically lowering the acceptance rate of the algorithm (i.e., its efficiency). **In fact, exploring an area is conditioned by the width of the proposal distribution, and adjusting it is the whole difficulty of the problem. Unless we turn to algorithms that guide the search.** For example, by using methods called HMC for Hybrid Monte Carlo (sometimes called Hamiltonian Monte Carlo)<sup>133</sup>.

132. Image taken from the notebook [https://github.com/jecampagne/cours\\_mallat\\_cdf/blob/main/2023/Monte\\_Carlo\\_Sampling.ipynb](https://github.com/jecampagne/cours_mallat_cdf/blob/main/2023/Monte_Carlo_Sampling.ipynb).

133. See a simple implementation in the notebook [https://github.com/jecampagne/cours\\_mallat\\_cdf/](https://github.com/jecampagne/cours_mallat_cdf/)

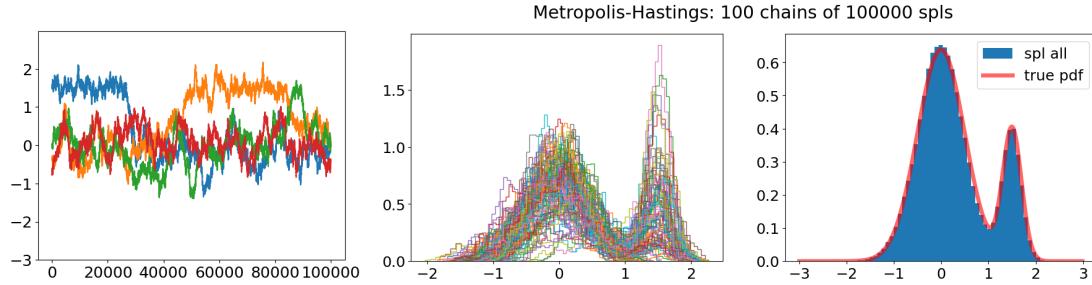


FIGURE 43 – For the same type of problem as in Figure 42, we used a sufficiently wide distribution to obtain the first sample and evolved 100 chains in parallel (we used 100,000 samples per chain after removing the first 25,000). Both modes of  $\pi(x)$  are well explored. Left: evolution of the first 4 chains. Middle: superimposed histograms of the 100 chains. Right: histograms of all accumulated samples compared to the distribution  $\pi(x)$ .

So, the moral, if you will, is that one should look at these convergence theorems with some skepticism. **Transitioning to high dimensions poses a problem.** There is a very elegant way, which we will see in the next session, to approach the problem differently with **Score-Driven algorithms**. The major difference is that we will use the data to modify the proposal distribution in a certain way to ensure faster convergence, even in high dimensions.

## 9. Session of March 13

During this final session, we will address sampling using the **Score Diffusion** method, which is the basis for algorithms implemented, for example, in language models such as GPT and Gemini (Sec. 2.1). Recall that the challenge is to sample from a Gibbs-type distribution in high dimensions:

$$p(x) = Z^{-1}e^{-U(x)} \quad x \in \mathbb{R}^d \quad (234)$$

---

blob/main/2003/Monte\_Carlo\_Sampling.ipynb, and the notebook [https://github.com/jecampagne/cours\\_mallat\\_cdf/blob/main/2003/Monte\\_Carlo\\_Sampling\\_2.ipynb](https://github.com/jecampagne/cours_mallat_cdf/blob/main/2003/Monte_Carlo_Sampling_2.ipynb) demonstrates its usage with a library. Also, see the article by Matthew D. Hoffman and Andrew Gelman (2014) for a variant called No-U-Turn <https://jmlr.org/papers/volume15/hoffman14a/hoffman14a.pdf>.

During the course, we have seen the expressions of  $U(x)$  giving rise to **structureless Gaussian distributions**, and then we have seen the case of more complex distributions in **Statistical Physics** where we used **Markov fields with locally conditional dependencies** (e.g.,  $\phi^4$  theory and Ising). All of this is primarily well understood in Physics, and we also manage it with neural networks. **The big surprise is that very large models have the ability to generate images of faces, environments in the broadest sense, etc. The challenge is to understand the mathematics behind these generative models.** There are certainly algorithms that we understand quite well, but the question is, as with the case of Stat. Phys., are we estimating an underlying probability distribution for faces, environments, etc.? If so, **what is the nature of these distributions?** Because in high dimensions, due to the curse of dimensionality, if we can generate faces, it means **there is some form of simplification of correlation structure**. Note that we can estimate these correlations with many examples, but they are few compared to what we would have expected if there had been an exponential explosion with dimension.

The idea of the Score Diffusion algorithm was seen in Markov chains (Fig. 44): we start from the distribution  $p(x)$  which is potentially very complex, and through **transport** (indexed by  $t \in [0, T]$ ) we simplify it to a distribution easy to sample from. In the case of the Score Diffusion algorithm, **the distribution  $p_T(x)$  is a Gaussian**. And the challenge afterwards is to **reverse the transport process** so that from a sample of  $p_T(x)$  we obtain a new sample of  $p(x)$ . **We will characterize  $p(x)$  by  $p_T(x)$  and all conditional probabilities involved during the inverse transport.**

Therefore, the framework is that of **Markov chains** (1906), but as it stands, this framework is too general. In particular, what type of chain should we consider? What does the "*forward*" chain refer to in Figure 44? The framework that will guide us is that of the **Renormalization Group** (see note 63) from the 1970s in Theoretical Physics (Statistical and Particle Physics), where the "time" axis  $t$  is a **scale axis**. In **generative AI** through Score Diffusion (2020), the "time" axis is that of **adding noise**.

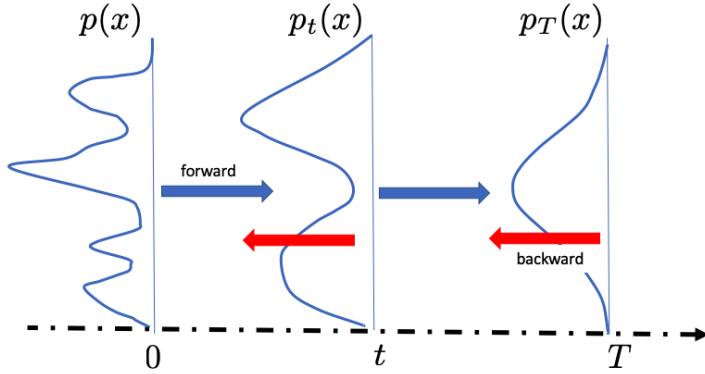


FIGURE 44 – Evolution of the distribution  $p(x)$  during the transformation steps indexed by  $t$  in the direction of adding noise (blue arrows), and if we can reverse the process (red arrows) then from a sample of  $p_T(x)$  we obtain a new sample of  $p(x)$ .

## 9.1 Ornstein-Uhlenbeck Equation

This concerns the evolution by transport ("forward") of the distribution  $p(x)$  (Fig. 44). Let  $x_t$  denote the *random variable* (which follows  $p_t(x)$ ) whose equation is given by

$$dx_t = -x_t dt + \sqrt{2} dB_t \quad t \in [0, T] \quad (235)$$

where  $dB_t$  is a Brownian motion. If we discretize it with a time step  $\delta$ :

$$x_{t+\delta} - x_t = -x_t \delta + \sqrt{2\delta} z \quad z \stackrel{iid}{\sim} \mathcal{N}(0, Id) \quad (236)$$

This is a Markov chain<sup>134</sup> that we have already studied (Sec. ??) which can be written as

$$x_{t+\delta} = (1 - \delta)x_t + \tilde{z} \quad \tilde{z} \stackrel{iid}{\sim} \mathcal{N}(0, 2\delta Id) \quad (237)$$

and when  $\delta \rightarrow 0$ , the solution at time  $t$  takes the compact form

$$x_t = e^{-t}x_0 + (1 - e^{-2t})^{1/2}z \quad z \stackrel{iid}{\sim} \mathcal{N}(0, Id) \quad (238)$$

<sup>134</sup> NDJE. see notebook [https://github.com/jecampagne/cours\\_mallat\\_cdf/blob/main/2024/Ornstein\\_Uhlenbeck.ipynb](https://github.com/jecampagne/cours_mallat_cdf/blob/main/2024/Ornstein_Uhlenbeck.ipynb)

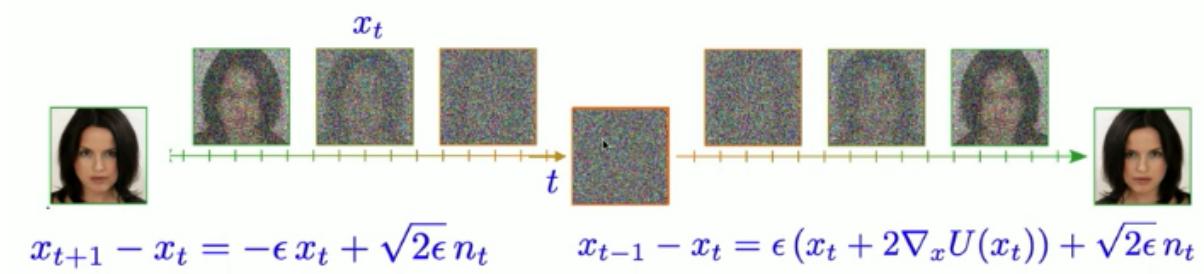


FIGURE 45 – Example of the evolution of an image by the stochastic process of a diffusion model: on the left progression where noise is injected (*forward*) and on the right where denoising is performed.

The proof is immediate. We notice the "forgetting of the initial distribution," and the convergence occurs exponentially towards Gaussian white noise. The illustration of this evolution of  $p(x)$  is shown in Figure 45 (it's the same as in section 2.7 reproduced here for readability).

What is the expression of the distribution of  $x_t$ ? In the "forward" mode, we know that we converge to Gaussian white noise. Let's consider the following renormalization of  $x_t$  which reveals the variance explosion as a function of time:

$$\tilde{x}_t = e^t x_t = x_0 + \underbrace{(e^{2t} - 1)^{1/2}}_{\sigma_t} z \quad (239)$$

We have an equation of the type where the "noisy signal" is the sum of the "original signal" and the "noise" whose variance grows exponentially. At any "time"  $t$ :

$$p_t(\tilde{x}_t) = \int p(\tilde{x}_t, x_0) dx_0 = \int p_t(\tilde{x}_t | x_0) p_0(x_0) dx_0 \quad (240)$$

where

$$p_t(\tilde{x}_t | x_0) = \frac{1}{(2\pi\sigma_t^2)^{d/2}} e^{-\frac{\|\tilde{x}_t - x_0\|^2}{2\sigma_t^2}} = g_{\sigma_t}(\tilde{x}_t - x_0) \quad (241)$$

We notice that we have a **convolution**:

$$p_t(\tilde{x}_t) = \int p_0(x_0) g_{\sigma_t}(\tilde{x}_t - x_0) dx_0 = (p_0 * g_{\sigma_t})(\tilde{x}_t) \quad (242)$$

Thus, we proceed with a progressive smoothing of the initial distribution  $p_0$  with a Gaussian.

Now, **we need to reverse the process** by calculating the reversed conditional probabilities. In the forward direction, we need  $p(x_{t+\delta}|x_t)$ , and in the backward direction, we need  $p(x_t|x_{t+\delta})$ , knowing that for continuous evolution by differential equation,  $\delta \rightarrow 0$ . While the existence of the inverse chain is not a problem in itself, its calculation is what catches our attention. We will provide arguments justifying that the variable  $x_{T-t}$  follows the following equation:

$$dx_{T-t} = (x_{T-t} + 2\nabla_x \log p_{T-t}(x_{T-t}))dt + \sqrt{2} dB_t \quad t \in [0, T] \quad (243)$$

This is a **damped Langevin equation** where the **score** (Sec. 5.4) appears. To understand this equation structure<sup>135</sup>, we will take the denoising perspective, which was not the viewpoint of those who emerged the solution via stochastic differential equations<sup>136</sup>.

## 9.2 Denoising Problem

The classic problem is that of observing  $y$  of a signal  $x$  tainted with additive noise  $z$  such that

$$y = x + z \quad z \stackrel{iid}{\sim} \mathcal{N}(0, \sigma^2 Id) \quad (244)$$

The challenge is to best recover  $x$  from  $y$ . To do this, we have an estimator of  $x$ , denoted  $\hat{x}(y)$ , which minimizes the mean squared error, i.e., minimizes

$$\mathbb{E}_{x,z} \|x - \hat{x}(y)\|^2 \quad (245)$$

---

135. NDJE. for interested people have a look at D. McAllester *On the Mathematics of Diffusion Models*, <https://arxiv.org/pdf/2301.11108.pdf>.

136. Yang Song et al. (2020) "Score-Based Generative Modeling through Stochastic Differential Equations", <https://arxiv.org/pdf/2011.13456.pdf>.

where  $x$  and  $z$  follow their respective laws.

Quick reminder: let  $x$  be a random variable, if we seek a constant  $\mu$  that best approximates  $x$  in mean squared error, the solution is

$$\min \mathbb{E}_x[\|x - \mu\|^2] \Leftrightarrow \mu = \mathbb{E}_x[x] \quad (246)$$

In our denoising case, we have an additional information, namely the knowledge of  $y$  (the observation) so:

$$\hat{x}(y) = \mathbb{E}_x[x|y] = \int x p(x|y) dx \quad (247)$$

**The problem of computing this optimal estimator** essentially began in the 1940s with the work of Wiener<sup>137</sup>. The problem in high dimension is the computation of  $p(x|y)$ , so we started by simplifying the search, for example, by trying to find a linear estimator in  $y$ , such that  $\hat{x}(y) = L.y$  where the operator  $L$  is optimized. Other classes of operators have subsequently been used.

But can we characterize the optimal solution, which is that of equation 247? Here is a proposition that will link with the previous section:

**Theorem 15 (Tweedie, Robbins, Miyasawa-1956-61)**

*The solution  $\hat{x}(y)$  is given by*

$$\hat{x}(y) = y + \sigma^2 \nabla_y \log p(y) \quad (248)$$

*where the score appears.*

**Proof 15.** The distribution  $p(y)$  can be written by revisiting the calculation of equation 242

$$p(y) = \int p(x) g_\sigma(y - x) dx \quad (249)$$

---

137. Norbert Wiener (1894-1964)



FIGURE 46 – Scheme of a denoiser consisting of a neural network whose parameters  $\theta$  need to be determined. As a by-product, the network provides an estimator of the score via  $\hat{x}(y) - y$  (up to the noise variance factor).

So, naturally and thanks to the Gaussian  $g_\sigma$ , it follows

$$\nabla_y p(y) = \int p(x) \left( -\frac{y-x}{\sigma^2} \right) g_\sigma(y-x) dx == \frac{1}{\sigma^2} \int (x-y) p(y, x) dx \quad (250)$$

Thus, by calculating the score, we have

$$\sigma^2 \nabla_y \log p(y) = \int (x-y) \frac{p(x|y)p(y)}{p(y)} dx = \int x p(x|y) dx - y = \hat{x}(y) - y \quad (251)$$

■

With this theorem, we understand why **the score of the noisy distribution appears in the backward process** (Eq. 243), which is essentially a form of denoising. But how do we estimate the *score*, especially since we are in high dimension? The big surprise is that we can achieve it with a neural network.

### 9.3 Denoising Network

The challenge is to estimate the term  $\nabla_x \log p_{T-t}(x_{T-t})$  (the score) in equation 243. The idea is to use a denoising network. Indeed, if we are able to obtain the optimal estimator  $\hat{x}(y)$ , i.e., the one that minimizes the quadratic risk and thus gives access to  $x$ , then we also estimate the score (Eq. 248).

So, the idea is to build a denoiser in the form of a neural network (Fig. 46). In a classical manner, starting from  $y$  the noisy observation, passing through the denoiser dependent on the parameters  $\theta$ , we obtain  $\hat{x}_\theta(y)$ . To determine the optimal value of  $\theta$ ,

we naturally use the quadratic risk that we minimize with samples  $\{x_i, y_i\}_{i \leq n}$  where  $y_i = x_i + z_i$  ( $z_i$  independent Gaussian white noise with variance  $\sigma^2$ ):

$$\theta^* = \operatorname{argmin}_{\theta} \frac{1}{n} \sum_{i \leq n} \|\hat{x}_{\theta}(y_i) - x_i\|^2 \quad (252)$$

Thus, an estimator of the score is obtained by

$$\hat{S}(y) = \frac{1}{\sigma^2}(\hat{x}_{\theta^*}(y) - y) \approx \nabla_y \log p(y) \quad (253)$$

The question that arises in the case of estimating  $\nabla_x \log p_{T-t}(x_{T-t})$  (for all  $t \in [0, T]$ ) is: what are the training samples? In fact, **we will use the forward phase** to train a "universal" denoiser, meaning that it can denoise without knowing the value of  $\sigma_t$  a priori. It is observed that this type of network performs better than if it were trained to denoise only for a single noise level. Can this phenomenon be explained?

Imagine that the signal  $x$  "lives" on a potentially very irregular manifold (in high dimension), then the probability density  $p_0(x)$  is a measure concentrated on this manifold. We have seen<sup>138</sup> that  $p_t(x_t)$  is a form of smoothing of  $p_0(x)$  by a Gaussian  $g_{\sigma_t}$  (Eq. 240). In the *backward* mode, initially at  $t = T$ , the value of  $\sigma_T$  is large, which gives  $p_T(x_T)$  an extremely regular shape. As  $\sigma_t$  decreases, we begin to see the details of  $p_0$ . Therefore, gradually, if we initialize a Markov chain with  $p_T(x_T)$  very smooth, the evolution of  $p_t(x_t)$  will drive the chain towards regions of high probability. This is illustrated<sup>139</sup> by a simple example in Figure 47. It is a homotopic deformation of the distribution  $p_T(x_T)$  from a white noise towards the potentially very complex distribution  $p_0(x_0)$ .

## 9.4 Transition from Memorization to Generalization

S. Mallat succinctly describes the typical architecture of convolutional networks used in this denoising framework, known as U-Nets<sup>140</sup> (Fig. 22). The first part of the network

138. Notebook reference is missing here, but it applies to  $x_t$  as well.

139. Notebook reference is missing here, but it's extracted from notebook [https://github.com/jecampagne/cours\\_mallat\\_cdf/blob/main/2024/ScoreDiffusionGene.ipynb](https://github.com/jecampagne/cours_mallat_cdf/blob/main/2024/ScoreDiffusionGene.ipynb).

140. Olaf Ronneberger, Philipp Fischer, and Thomas Brox (2015) developed such an architecture for a medical image segmentation problem. See <https://arxiv.org/abs/1505.04597>.

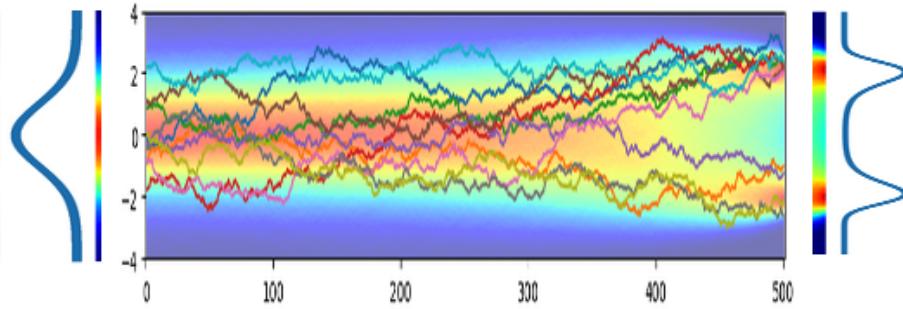


FIGURE 47 – Evolution of Markov chains in the backward mode starting from a Gaussian distribution towards a distribution with 2 modes.

performs convolution and downsampling operations while the second part carries out deconvolution and upsampling operations to restore the original image. He also provides images generated by generative models that everyone has seen here and there in the press, such as those of celebrities who "do not exist," as shown in Figure 48. Other types of images, such as bedrooms, are also well generated after training. Finally, we have images generated from a natural language query, such as the one in Figure 1 of Section 2.2.

The real question when we see such images produced by these generative models is: **are we sampling a probability density?** In other words, has the model learned  $p(x)$  with  $x$  being a face image, and what is the nature of this distribution  $p$ ? The alternative would probably be that the new images are sophisticated patchworks of images from the database? *Note. S. Mallat gave us some elements of the answer in his first session this year (Sec. 2.7 Fig. 8), and he reformulates them in this session.* The study consists of training two generative models with two disjoint sets of size  $N$  from the same database of face images. Then, we generate an image with both models from the same initial white Gaussian noise image. The result is as follows: as the size  $N$  of the training sets increases, we transition from a mode of *memorization* to a mode of *generation*. This is illustrated<sup>141</sup> in Figure 49. **We have indeed learned the deterministic inverse transport** which allows us to generate a new independent face image from any new realization of

---

141. Source: Zahra Kadkhodaie, Florentin Guth, Eero P. Simoncelli, and Stéphane Mallat (2024) *Generalization in diffusion models arises from geometry-adaptive harmonic representations* <https://arxiv.org/pdf/2310.02557.pdf>.



FIGURE 48 – Some examples of face images generated by a generative model trained with a large database using a score diffusion method.

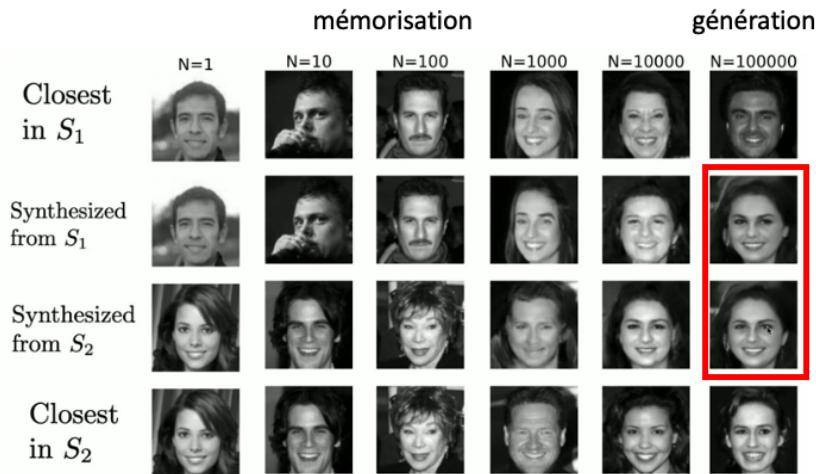


FIGURE 49 – Two generative models  $S_1$  and  $S_2$  are trained with two non-overlapping databases, each of size  $N$ . The same noise image is then given to both models to synthesize a new image ( $80 \times 80$ ). We can find the image from the database that best approximates this synthesized image. As long as  $N < O(10,000)$ ,  $S_1$  and  $S_2$  produce different images that increasingly resemble an image from the database as  $N$  decreases. Once  $N = 100,000$ , the images synthesized by  $S_1$  and  $S_2$  are (almost) identical, but different from the images closest to the two databases. We indeed witness a transition towards a model that learns a probability distribution.

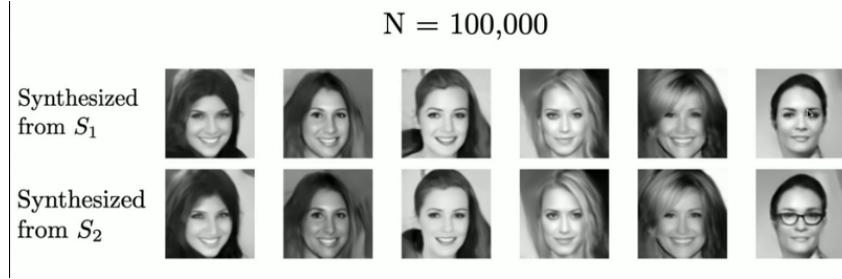


FIGURE 50 – Synthesized images from models  $S_1$  and  $S_2$  (Fig. 49) trained with  $N = 100,000$  images, where the same white Gaussian noise image is given at the beginning of each generation. The vast majority of the synthesized images are identical with minor instabilities.

white noise, independent of the initial database<sup>142</sup>. If we repeat the image synthesis by the two models trained with 100,000 images, providing them with the same initial noise image each time, we obtain the same generated images, except for minor details reflecting the differences between the two optimized networks (Fig. 50). The same type of experiment was conducted with other types of images (e.g., bedrooms). The conclusions are the same.

The transition from memorization to generation around  $N = 100,000$  depends on the size of the neural network learning the score and the resolution of the images: in the experiments conducted above, the network was a U-Net with  $7 \times 10^6$  parameters and the images were of size  $80 \times 80$  pixels. If we have images of size  $40 \times 40$ , we need a smaller network, and generalization is effective when  $N = O(10,000)$ . In essence, this is a classic explanation of interpolation: the more parameters we have, the more training images we need. What is more surprising is that if we estimate the size of the variety where the bedrooms are located, we intuit a dimension of the order of  $\log_2(10^5) \approx 17$ , which is relatively modest compared to the number of pixels ( $80^2 = 6,400$ ). So, one might think that, after all, all bedrooms are almost identical in composition with minor details. This leads S. Mallat to reflect more generally on the perception we may have of the diversity of the world: in a way, **are we not overestimating the complexity of the world?** And if, after all, the world is simpler than it seems, then this would explain why neural networks are able to capture the correlations that structure these images.

<sup>142</sup>. At least to some extent because if we only take one type of Hollywood celebrity face, we will not generate a face of an Asian, African, etc.

## 9.5 Opening towards some research directions

S. Mallat develops research directions and continues his session by projecting slides (*Note. around 56:15 after the start of the video.*). These are themes he will address in 2025.

### 9.5.1 Return to denoising: wavelet bases

The problem of denoising, as mentioned earlier, dates back to the 1940s. *Note. It is also worth noting that the 2021 course covered the classic viewpoint of sparse representations which led to data compression, but could just as well have covered the aspect of denoising.* In fact, let's take an orthonormal basis  $\{\psi_k\}_k$  and decompose the noisy signal  $x_t = x + z$  ( $z \sim \mathcal{N}(0, \sigma_t^2 Id)$ ):

$$x_t = \sum_k \langle x_t, \psi_k \rangle \psi_k \quad (254)$$

How can we remove the noise component? In fact, since the decomposition is linear:

$$\langle x_t, \psi_k \rangle = \langle x, \psi_k \rangle + \langle z, \psi_k \rangle \quad (255)$$

But  $z$  being a Gaussian random variable, the same is true for  $\langle z, \psi_k \rangle$ , which will generate small coefficients (at least if  $\sigma^2$  is not too large compared to the signal). Therefore, applying a **thresholding via a non-linearity (ReLU)** (Fig. 51) allows us to keep only the large coefficients that are expected to be sufficient to reconstruct a good approximation of the signal:

$$\hat{x} = \sum_k \rho_T(\langle x_t, \psi_k \rangle) \psi_k \quad \text{with} \quad \rho_T(u) = \max(u - T, 0) \quad (256)$$

Typically, one chooses  $T \approx (3 \div 5)\sigma$ .

This is an **adaptive denoising algorithm** developed in the 1980s described in the 2021 course. It is *adaptive* because it adapts to the signal decomposition; in comparison, Fourier denoising would impose a cutoff on  $k < k_{\max}$ , which in context would be a frequency index.

The challenge is to find **the right representation basis** to achieve the most par-

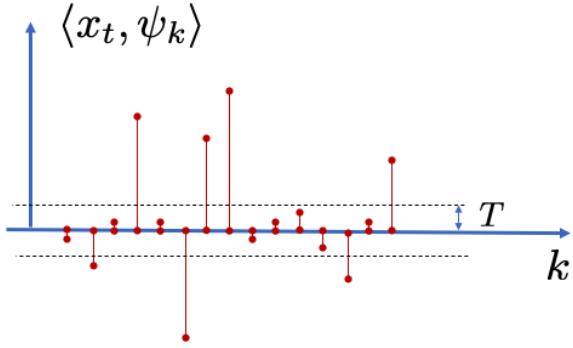


FIGURE 51 – Example of thresholding of scalar products to eliminate noise, which generates small coefficients compared to the few large coefficients that are sufficient to reconstruct a good approximation of the signal.

simonious signal decomposition possible. In this context, in the 1990s, **multi-resolution analyses with wavelet bases** were developed<sup>143</sup>. The interest of wavelets is to obtain parsimonious representations of the signal whose coefficients are large only at discontinuity zones. Imagine an image of an object such as a vase on a plain background; the coefficients are localized at the edges of the vase. The same goes for the contours of a face, the eyes in a face, etc. This applies to all spatial scales and areas of an image. Therefore, we have a well-suited representation for shape recognition and identification, even in the presence of noise. An example of denoising on a 1D signal is shown<sup>144</sup> in Figure 52. Notice that denoising did not involve averaging over a sliding window or eliminating high-frequency Fourier components, for example. **We retained all the discontinuities of the noise-free signal.** The residual artifacts at the discontinuities resemble a Gibbs phenomenon but are greatly reduced compared to its Fourier counterpart. The generalization to 2-dimensional orthonormal wavelet bases is almost immediate, and thresholding of coefficients can also be performed to achieve equally effective denoising.

143. NDJE. I invite you to refer to the 2021 course, which covers what S. Mallat briefly mentions in these slides this year

144. NDJE. I used the `swt/iswt` functions from the `pywavelet` library, with `level=6` for a 2048-sample signal and "db2" as the wavelet.

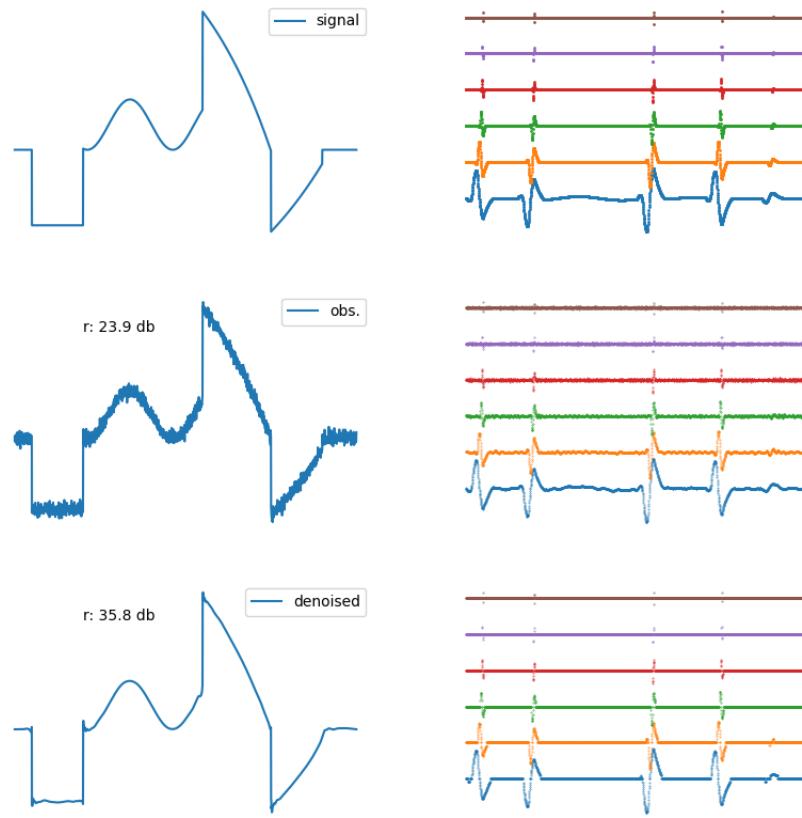


FIGURE 52 – Example of denoising thresholding by simple thresholding at  $T = 5\sigma$  on wavelet coefficients. Left: the signal without noise (top), the signal with added Gaussian white noise  $\sigma = 3$  (middle), the reconstructed signal after thresholding. Right: the detail coefficients of the three signals at different scales (refining further upwards). The metric used is  $r = 10 \log_{10}(\|x_r\|^2 / \|x - x_r\|^2)$  where  $x_r$  is the noise-free signal taken as reference.

### 9.5.2 Denoising networks: What do they do?

What is the connection with what neural networks do? It has been seen that performing optimal denoising (Th. 15) is to estimate the score of a distribution:

$$\hat{x}(x_t) = x_t + \sigma_t^2 \underbrace{\nabla_{x_t} \log p(x_t)}_{\hat{s}(x_t)} \quad (257)$$

For this purpose, we will use a universally optimized denoising network during the forward phase (Sec. 9.3) such that:

$$\hat{x}(x_t) = \hat{x}_{\theta^*}(x_t) \quad (258)$$

However, if the neural network uses only ReLU and all bias terms have been removed<sup>145</sup> then it is *locally homogeneous of degree 1*, and according to Euler's theorem<sup>146</sup>, it follows:

$$\hat{x}_{\theta^*}(x_t) = \nabla_x(\hat{x}_{\theta^*}(x_t)) \cdot x_t \quad (259)$$

where the Jacobian of  $\hat{x}_{\theta^*}$  appears. We can diagonalize<sup>147</sup> it in an orthonormal basis of eigenvectors denoted  $\{\psi_k\}_k$  associated with eigenvalues  $(\lambda_k)_k$ . Projecting  $x_t$  onto this basis, we can then write:

$$\hat{x}(x_t) = \sum_k \lambda_k \langle x_t, \psi_k \rangle \psi_k \quad (260)$$

Notice that if  $\lambda_k \approx 0$ , the coefficient  $\langle x_t, \psi_k \rangle$  is canceled. **Thresholding of the coefficients of the decomposition of  $x_t$  via the eigenvalues is performed.** However, unlike the case of denoising by thresholding wavelet coefficients, here **the orthonormal basis is not chosen a priori, it comes from the diagonalization of the Jacobian, it will therefore change**

145. NDJE. "bias-free" network: all additive constants of convolution operations and batch normalization operations are removed (i.e., batch normalization does not subtract the mean).

146. NDJE. A homogeneous function of degree  $k$  from  $\mathbb{R}^d$  to  $\mathbb{R}$  is characterized by

$$f(tx_1, tx_2, \dots, tx_d) = t^k f(x_1, x_2, \dots, x_d)$$

and Euler's theorem states that

$$kf(x_1, x_2, \dots, x_d) = \sum_{i=1}^f x_i \frac{\partial f}{\partial x_i}((x_1, x_2, \dots, x_d))$$

147. NDJE. Using singular value decomposition, SVD.

depending on the input signal  $x_t$ .

One can at this stage pose a number of questions regarding the probability estimation problem:

- The first is **dependence on the data**, whether there is invariance or not of the result with respect to the (training) dataset? According to the numerical experiment on generating images of faces and bedrooms, it seems that **there is a certain independence** from the training dataset (*NDJE. with the caveat that not all types of faces or all types of existing bedrooms in the world are generated*).
- But let's use a (trivial) argument: one can always generate an image filled with 0 regardless of the data, so the result is by construction independent of the data. Thus, it is not sufficient to say that the result is independent of the dataset. Therefore, we must **question the precision of the result**.

S. Mallat talks about "variance" for the first and "bias" for the second. We can understand variance as follows: if the result depends on the dataset, changing it modifies the result, so there is a fluctuation of the result induced by the choice of dataset. Fluctuation implies variance. As for bias, it concerns the precision with which you will obtain the result<sup>148</sup>.

So, the question is **whether we have correctly estimated**  $\nabla_x \log p(x) = -U(x)$ ? In the face/bedroom generation experiment, it seems at first glance that the generated samples resemble what one would expect. Can we have a mathematical confirmation of this visual impression? Here, one must be able to find/extract a mathematical question that captures the essence of the problem one seeks to demonstrate without it being too simple but still accessible.

### 9.5.3 Results of numerical experiments

In this case, we will use simpler images like the one in Figure 53 where the background has  $\alpha$  derivatives just like the separating line between two regions. Such images,

---

148. NDJE. It is important to contextualize the usage of vocabulary. This terminology may be perceived differently. When making a measurement in Physics, one indicates the statistical error (variance) which is due to the number of samples collected to perform the measurement, and the systematic error (bias) related to the way the measurement is carried out, estimating efficiencies, estimating theoretical calculations of extracted quantities, etc. (there can be identified several types of systematics including those dependent on statistics such as simulations to estimate a detection efficiency).

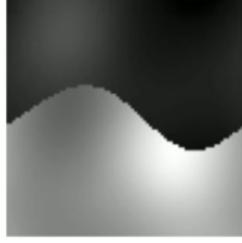


FIGURE 53 – Image of type  $C^\alpha$  where we have a smooth curve on a smooth background controlled by  $\alpha$ . For example,  $\alpha = 4$ .

S. Mallat tells us, were extensively studied in the 2000s because at that time the challenge was to show that "traditional" wavelet denoising techniques were not optimal due to an inability to adapt to **the geometry of the problem**. Let's simply say that the supports of the wavelets, although localized, are like blocks whose deformations do not adapt to the geometry of the transition curve. **We would like wavelets whose support can follow the contours.** This led to a plethora of works including the construction of "bandlet" bases<sup>149</sup>. The idea is therefore to understand how to construct optimal estimators of the noisy signal. Thus, starting from these images  $x$ , we apply noise  $z \sim \mathcal{N}(0, \sigma^2)$ , and we denote  $x_t = x + z$ . It is demonstrated that the optimal estimator  $\hat{x}$  has an error such that

$$\|\hat{x} - x\|^2 \sim \sigma^{2\alpha/(\alpha+1)} \quad (261)$$

when thresholding the coefficients in the bandlet basis. At the time, S. Mallat tells us, achieving the optimum was difficult, and we didn't quite get there (there was a  $\log \sigma$  factor). But this provides a well-defined mathematical framework. The question is: what will the U-Net type neural network do? Is it capable of achieving the optimum?

The results<sup>150</sup> are shown in Figure 54. First (top figure), we can observe the vectors of the orthogonal basis of the Hessian. They are indexed by their eigenvalues, and what we observe is that 1) we see kinds of oscillating functions in the two zones of the image (top/bottom) with zero means on either side of the boundary, 2) we can distinguish a

---

149. NDJE. See for example Ch. Dossal, E. Le Pennec, and S. Mallat (2011) <https://www.di.ens.fr/~mallat/papiers/2011-SigPro-DLPM.pdf> as well as works done with G. Peyré.

150. NDJE. Extracted from Z. Kadkhodaie, F. Guth, E. P. Simoncelli, S. Mallat (2023-24) <https://arxiv.org/abs/2310.02557>.

narrow band following the boundary. It is important to note that the oscillating functions compute variations over regular regions and not variations across the boundary because that would lead to large coefficients due to the discontinuities contaminating all other coefficients<sup>151</sup>. **All this tells us that these eigenvectors indeed form an optimal basis** that we knew mathematically and that the network has learned with image denoising. Then (bottom figure), we can compare the denoising speed: that is, the evolution of the function giving the PSNR<sup>152</sup> of the denoised image as a function of the PSNR of the noisy image. We only control the theoretical slope. We observe that **the network behaves in an equally optimal manner**.

Another example is shown in Figure 55: the base image is a disc with variable radius and position, as well as colors (gray levels) inside and outside the disc. Therefore, we have a 5-dimensional variety of the tangent space in which the images exist. We can calculate the 5 eigenvectors (top figure). In the bottom figure, we see that the network also learns 5 eigenvectors that closely resemble those calculated. However, it may add other suboptimal components, but they still adhere well to the previously discussed characteristics.

Therefore, it seems that for cases where we can calculate the optimal eigenvectors, we can confirm that the network **denoises optimally**, thus **estimating the score and therefore the probability optimally**. What does it do with face images? An example is provided in Figure 56. **What is particularly remarkable is the adaptation of the orthogonal basis to the geometry of the image with oscillating shapes in uniform areas that respect the contours.**

#### 9.5.4 Reflections on the generative model by score diffusion

It can be observed that the neural network (e.g., U-Net) used to estimate the score is capable of estimating the underlying probability of face images, bedrooms, etc. It is capable of **generalization**, but as shown in Section 9.4, **it requires a lot of examples**. For example, in the numerical experiments shown in the previous sections, it took 100,000 images to ensure that the model could truly generate new  $80 \times 80$  images. So, there is a problem, either in cases where larger realizations are needed, or in cases where there are not many images available, even if they are of small dimensions. In particular, in the

---

151. NDJE. This is a clarification provided by S. Mallat after his lecture.

152. NDJE. It's a metric that estimates the similarity between two images, *Peak Signal to Noise Ratio*.

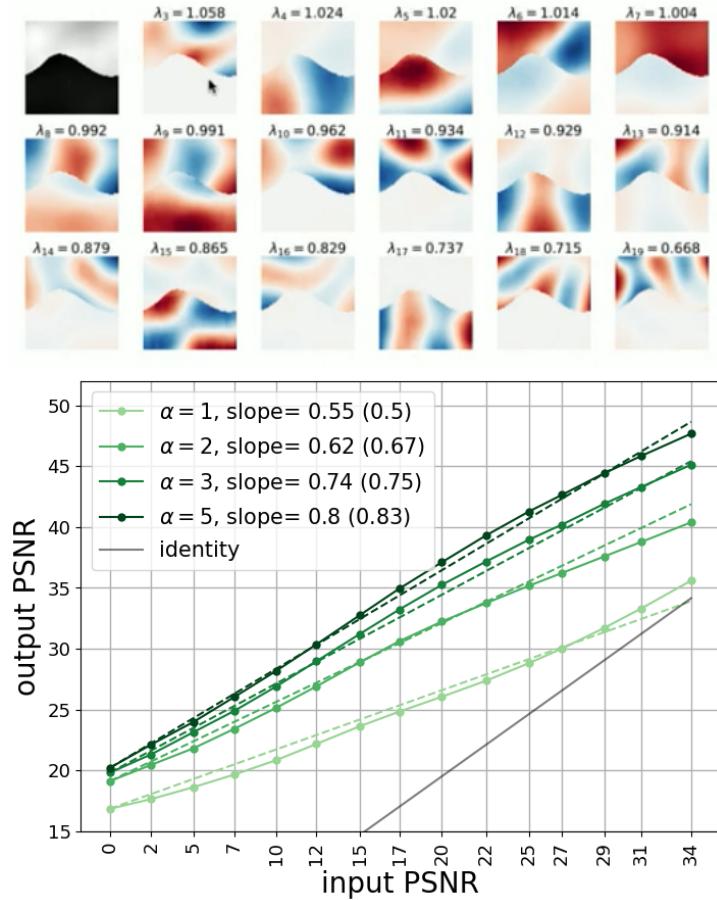


FIGURE 54 – Top: Image 53 ( $80 \times 80$ ) and eigenvectors of the largest eigenvalues. Bottom: Comparison of theoretical curves (dashed lines) and experimental measurements (points) of the evolution of the PSNR of the denoised image (output) as a function of the PSNR of the noisy image (input).

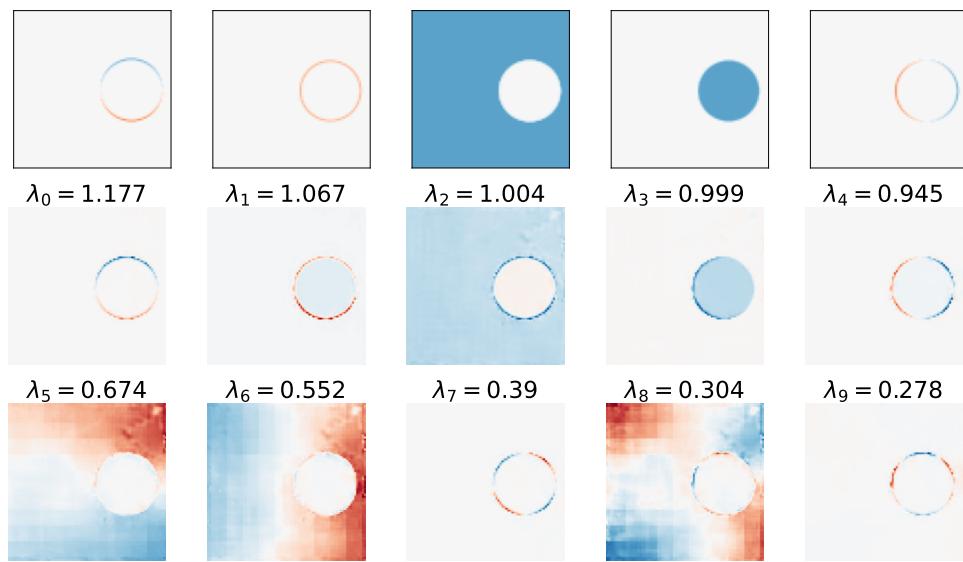


FIGURE 55 – Top: The 5 eigenvectors of the image space consisting of a disc with variable radius and position, as well as colors (gray levels) inside and outside the disc. Bottom: The first 5 eigenvectors of the Hessian obtained during denoising, which are identical to the expected ones, and then some components found by the network to complete the basis in the image space it used during training.

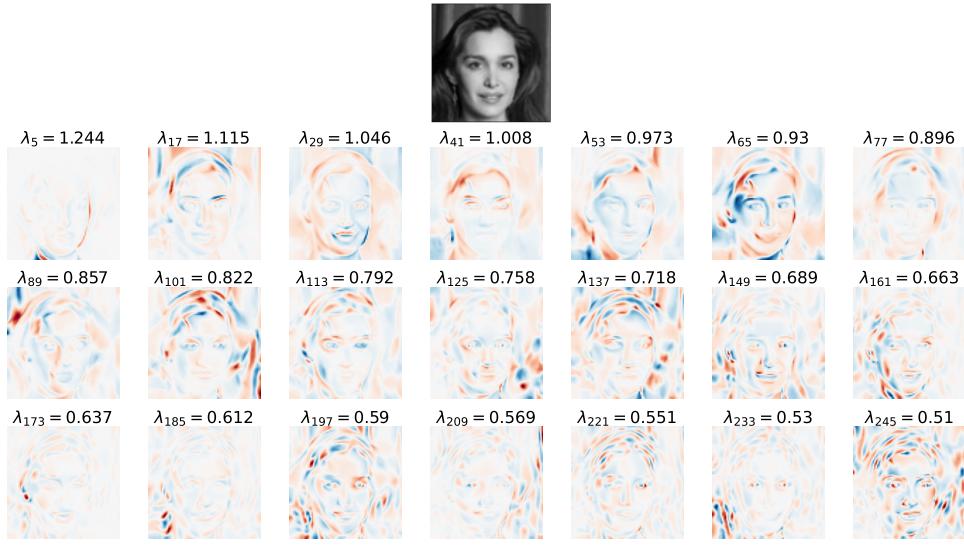


FIGURE 56 – Eigenvectors of the Hessian found by the network trained with noisy face images.

latter case, one will inevitably end up with **overfitting**, meaning that one is in the realm of **memorization**, which is not the intended goal.

Now, even though a large amount of data is needed, their number does not follow an exponential law with respect to the dimension, the model has somehow absorbed the structure of the data. What is it? To answer this question, one needs to open the "black box" (Fig. 22) and **understand the architecture** that carries meaning. This introspection is all the more important, S. Mallat tells us, when faced with a problem where data is limited, we need to build low-dimensional models<sup>153</sup>, and for that, one must understand the architectures.

### 9.5.5 Case study: turbulence

The problem proposed by S. Mallat is the characterization of turbulence. It is a subject that A. N. Kolmogorov laid the foundations for in 1941 but which remains open to this day. In fact, turbulence is the result of the dissipation at small scales of the energy

---

153. NDJE. See, for example, the reflections in Section 2.1.3 of the 2020 course.

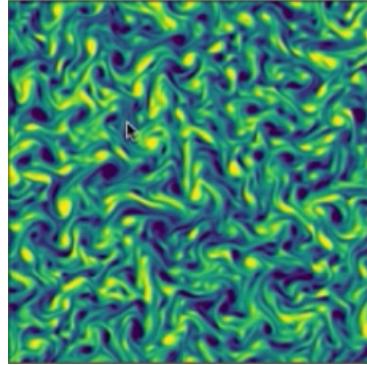


FIGURE 57 – Example of an image of turbulent fluid.

injected at large scales (effects of viscosity at high Reynolds numbers). Therefore, we want to be able to estimate the probability density in the form of Gibbs energy by considering the phenomenon as if it were in a stationary equilibrium (which is not strictly accurate). Nevertheless, the question is: what is the energy of a turbulent fluid in 2D? When we look at an image of turbulent fluid (Fig. 57), we notice the numerous swirls, filaments, and eddies, etc.: **there is geometry**.

Now, by studying this phenomenon, Physics provides a possible **reduction of the number of degrees of freedom**: we are dealing with a **multi-scale structure** similar to that shown in Figure 4. Therefore, we can reduce from  $d$  direct interactions to  $O(\log d)$  multi-scale terms. The catch is that the groups at each scale level interact with each other, making this problem non-trivial.

To solve this type of problem, Physics (1970) developed the following scheme (Fig. 58):

Forward : initially, the image (or field in general)  $x$  is progressively averaged and downsampled to obtain versions  $x_j$  at different resolution scales  $j \in \{0, J\}$ . At the same time, we can observe the evolution of the probability distribution  $p_j(x_j)$  (for example, by fitting a linear energy model discussed in Section 6.5).

Reverse : then, we want to generate new images at the original scale. For this purpose, we notice that at scale  $J$ , the size of the image is reduced to  $(N/2^J)^2$ , which can be just a few pixels. Therefore, it is *a priori* easy to sample such a distribution in low dimension. Then, with a cascade in the "forward" step, we can compute the reverse

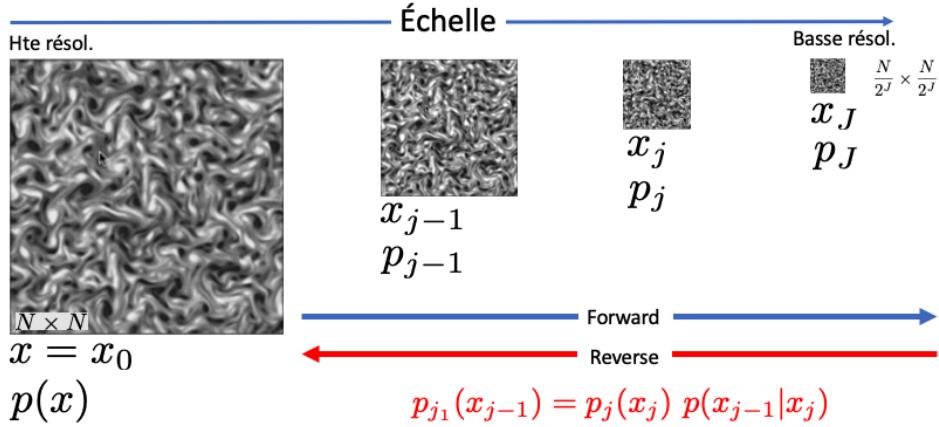


FIGURE 58 – Evolution scheme across scales: in the "forward" direction where the original image is progressively averaged and downsampled, and in the "reverse" direction where the Markov chain is reversed by starting with a sample from the low-dimensional probability (easy), and using the relationship between probabilities.

cascade. In fact, we have

$$p_{j-1}(x_{j-1}) = p_j(x_j) p(x_{j-1}|x_j) \quad (262)$$

**Instead of having a Markov chain evolving with the noise level as in the Score Diffusion method, here the chain evolves according to the scales.** This is somewhat the philosophy behind U-Net.

The point to clarify is how we go from scale  $j - 1$  to scale  $j$ ? This is where the wavelet transform will guide us, because to construct an image of resolution  $j - 1$  (higher) from an image of resolution  $j$  (lower), we need to **add "details"** denoted  $\bar{x}_j$  (Fig. 59). Now, the decomposition of  $x_{j-1}$  into  $(x_j, \bar{x}_j)$  is orthogonal, so we will assume that

$$p(x_{j-1}|x_j) = p(\bar{x}_j|x_j) \quad (263)$$

And just like we saw for Brownian motion (Sec. 4.3, Fig. 26) that **we can obtain long-range dependencies between variables through local but hierarchical dependencies (multi-scales)**, the same applies here. Now, to model  $p(\bar{x}_j|x_j)$ , we can use a linear Gibbs energy

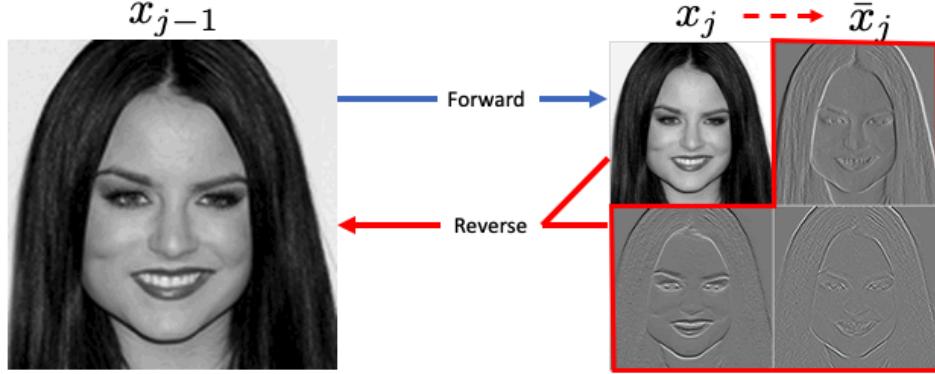


FIGURE 59 – When an image  $x_{j-1}$  passes through a 2D wavelet transform (forward mode), we obtain not only the averaged and downsampled image  $x_j$  but also the "detail" components denoted  $\bar{x}_j$ . In the "reverse" mode, where from an image  $x_j$  we want to produce an image  $x_{j-1}$  of higher resolution, we need to be able to construct the "detail" components.

model (Sec. 6.5):

$$p_{\theta_j}(\bar{x}_j|x_j) = Z_j^{-1} e^{\theta_j^T \Phi(\bar{x}_j, x_j)} \quad (264)$$

where  $(\theta_j)_j$  is of low dimension. This modeling is a completely open research challenge, as stated by S. Mallat.

Once the optimization is done, the path is clear. We start with a sample  $x_J$  from  $p_J$ , then we generate the detail patches  $\bar{x}_J$  according to  $p(\bar{x}_J|x_J)$ , and we reconstruct  $x_{J-1}$ . Again, we generate the patches  $\bar{x}_{J-1}$  according to  $p(\bar{x}_{J-1}|x_{J-1})$ , and we can reconstruct  $x_{J-2}$ , and so on. In this way, we end up with a sample  $x_0$  (or  $x$ ) from the probability

$$p(x) = p_{\theta_J}(x_J) \prod_{j=J}^1 p_{\theta_j}(\bar{x}_j|x_j) \quad (265)$$

With such models, we can proceed<sup>154</sup> to the generation of non-Gaussian fields by wavelet models like those in Figure 6 (*turbulence, cosmic web, etc.*).

<sup>154</sup>. NDJE. see Sihao Cheng, Rudy Morel, Erwan Ally, Brice Ménard, Stéphane Mallat (2023), <https://arxiv.org/abs/2306.17210>,, and Florentin Guth, Etienne Lempereur, Joan Bruna, Stéphane Mallat (2023) <https://arxiv.org/abs/2306.00181>, and the referenced works.

## 9.6 Conclusion of this year

The challenge is to **understand the structure**, because we want to know why neural networks are able to calculate scores and thus generate images of faces, bedrooms, etc., and in many other domains. For these problems, we don't have the slightest basis of a theoretical model as we can obtain with models in Physics where we have one or even several centuries of reflections that have provided these bases on which we can rely. But if the networks are able to capture these structures, as already stated, it may be that there is hope that this structure is not so complex after all, and **there may be some guiding principles** that could put us on the path to **attacking the problem mathematically**.

In the case of **score-based generative diffusion**, we try to show that we can indeed be in a mode of generalization (and not just data memorization). However, the algorithms rely on a "**black box**" namely the denoising network (e.g., U-Net). So if we want to go further in understanding, we need to open the said box.

Probably, one of the guiding principles, S. Mallat tells us, is undoubtedly the **notion of hierarchy**. In the case of an image as shown in the previous section, it is about scales. **In the case of an image**, it is much simpler, because the topology is well defined, the notion of dilation/contraction is well defined, which gives access to **the notion of scale**. On the other hand, imagine the **context of a company** where actors interact to carry out a project, there is no notion of natural scale, but certainly **the notion of hierarchy is present**. In this context, the article by H. Simons from the 1960s<sup>155</sup> provides a sort of explanation for this universality of hierarchical systems: there is a dynamic search for stability. However, a hierarchical structure is not a simple vertical tree-like structure with disconnected horizontal levels that would correspond to the stage of wavelet decomposition, for example. But, to take again the case of a company, **we want communication at each level of the hierarchy**. Similarly, in images of faces (turbulence, etc.), we see that there are **long-range interactions at all scales**.

**These connections pose again open questions in mathematics.** And undoubtedly, paths to understand the principles underlying these "horizontal" hierarchical structures are necessary. Because hierarchical systems in the form of simple trees have been tested in a whole range of domains: e.g., in linguistics with Noam Chomsky's grammars<sup>156</sup> but

---

155. NDJE. See Course 2020 Sec. 3.2

156. NDJE. See Course 2019 Sec. 2.3.1

it is clear that this is not what led to the large language models (see Introduction of this year); similarly in imaging, multi-resolution analyses (wavelets) allow for processing but fundamentally this is not what allowed image generators, similarly in audio etc.

In a way, if we draw a parallel, self-similar structures are simple forms of fractals but there are much more complex objects in the family of fractals, the same goes for hierarchies: those in simple tree-like structures have allowed us to take a step towards understanding hierarchical structures, but there is still a long way to go that we will continue to explore next year.

## 10. NDJE. Quelques ajouts personnels

In this section, I will add some elements related to the course:

- Concerning a widely used sampling technique called HMC.
- Regarding what is known as *Normalizing Flows*.

### 10.1 NDJE. Hybrid/Hamiltonian Monte Carlo

In this section, I will introduce a Monte Carlo method (denoted HMC) that aims to address the question: can we reduce the time required for a Markov chain to provide a batch of independent samples from a target distribution  $\pi(x)$ ? Throughout this section, I only touch on some concepts, and I refer, for example, to Michael Betancourt's article<sup>157</sup> (2018) for more details on this topic.

During this year's course, we have seen that we aim to sample probabilities such that  $\pi(x) > 0$ , which can be modeled as Gibbs distributions:

$$\pi(x) = Z^{-1} e^{-U(x)} \Leftrightarrow -\log(\pi(x)) = U(x) + \text{Const} \quad (266)$$

We will assume that we can compute the score,  $\partial U(x)/\partial x$ <sup>158</sup>.

---

157. Michael Betancourt, *A Conceptual Introduction to Hamiltonian Monte Carlo*, <https://arxiv.org/pdf/1701.02434.pdf>

158. Note that automatic differentiation libraries (e.g., TensorFlow, PyTorch, JAX) used for machine learning allow for efficient and stable codes if the expression for  $\nabla_x U(x)$  is not known analytically.

In the HMC technique, we introduce a conjugate variable to  $x$ , conveniently denoted as  $p$ , and we form the function (Hamiltonian):

$$H(x, p) = U(x) + E_k(p) \quad (267)$$

where  $E_k(p) = \|p\|_2^2/2$  represents a sort of *kinetic energy* of the system when  $U(x)$  represents its potential energy. But one should not push the analogy further. The dynamic equations of the system are then identical to those of classical Hamiltonian mechanics:

$$\dot{x} = \frac{\partial H}{\partial p} = \frac{\partial E_k}{\partial p} = p \quad \dot{p} = -\frac{\partial H}{\partial x} = -\frac{\partial U}{\partial x} \quad (268)$$

where the notation  $\dot{a}$  denotes a "temporal" derivative of the variable  $a$  ( $\dot{a} = da/dt$ ), and when time is discretized,  $\dot{a} = a(t+1) - a(t)$ .

The HMC algorithm revolves around the following simplified version to discuss its components. Starting from a state  $x_i$  of a chain to obtain the state  $x_{i+1}$ , we proceed as follows:

**Step 1:** Random draw of  $p_i \sim \mathcal{N}(0, 1)$ ; then initialize  $p_{new} = p_i$  and symmetrically  $x_{new} = x_i$ ;

**Step 2:** Iteration of  $n_{steps}$  steps to integrate the equations of motion following the method called the "leapfrog" algorithm:

$$\begin{cases} p_{new} &= p_{new} - \varepsilon \times \frac{1}{2} \frac{\partial U(x)}{\partial x} |_{x=x_{new}} \\ x_{new} &= x_{new} + \varepsilon \times p_{new} \\ p_{new} &= p_{new} - \varepsilon \times \frac{1}{2} \frac{\partial U(x)}{\partial x} |_{x=x_{new}} \end{cases} \quad (269)$$

Note that  $-\nabla_x U(x)$  acts as a "force", so if  $\varepsilon$  is a sort of infinitesimal time element  $dt$ , then we understand that  $dp = -dt \times \nabla_x U(x)$  (the  $1/2$  is there to account for the fact that we divide the time interval into two equal parts). Similarly,  $p$  (if the mass is unity) is a velocity, so  $dx = dt \times p$ . Therefore, we understand that we integrate the equations of motion. We note the update of the gradient along the way, which will come into play later;

**Step 3:** Then, we reverse the momentum  $p_{new} = -p_{new}$  (not necessary in practice, we will see);

**Step 4:** Finally, we proceed as for the Metropolis algorithm according to the acceptance probability of the new state  $(x_{new}, p_{new})$ :

$$p_{acc} = \min \left\{ 1, r = \frac{\tilde{P}(x_{new}, p_{new})}{\tilde{P}(x_i, p_i)} \right\} \quad (270)$$

with

$$\tilde{P}(x, y) = \exp\{-H(x, p)\} = \exp\{-U(x)\} \exp\{-E_k(p)\} \quad (271)$$

This is done by uniformly drawing a number  $u$  from the interval  $[0, 1]$ . If  $u < r$ , we accept  $x_{new}$  as the value for  $x_{i+1}$ , otherwise, we take  $x_i$ . Notice the decoupling between the two variables  $x$  and  $p$ . Also, note that we do not keep  $p_{new}$  afterwards.

The method raises the following question: why use Hamiltonian equations? The arguments are as follows:

1. The temporal *reversibility* of Hamiltonian dynamics is important for the reversibility of the Markov chain (see later);
2. Then, the *conservation* of the Hamiltonian over time means that in principle  $r = 1$ , and thus favors the acceptance of a new state of the chain. However, in practice, the conservation of  $H$  is not exact, even by the imperfection of the integration algorithm;
3. The proposal of the new state is guided by  $p_{new}$ , which points in the *direction of the highest probability density*, which is favorable;
4. Finally, Hamiltonian dynamics *preserves volumes in phase space*, this is Liouville's theorem. This property is crucial for being able to use the Metropolis method (and not the Hastings version) (step 4) to accept or reject a new state of the chain.

Now, two questions naturally come to mind: why does the algorithm (HMC) converge to  $\pi(x)$ , and why do we have this strange structure of the leapfrog algorithm, especially the momentum reversal at the end? The first question refers to the notion of *detailed balance*, which compares the probabilities of reverse processes  $A \rightarrow B$  and  $B \rightarrow A$  (See Sec. 11)<sup>159</sup>. So, let's see why we have the following equality which will guarantee that

---

159. Note that there are versions of HMC that do not satisfy this criterion because it is not necessary to obtain a stable distribution to which the algorithm converges: see for example <https://arxiv.org/pdf/1409.5191.pdf>.

$\pi(x)$  is the invariant measure of the algorithm:

$$\pi(x)p_t(x \rightarrow y) = \pi(y)p_t(y \rightarrow x) \quad (272)$$

Here,  $p_t(x \rightarrow y)$  is the transition probability from position  $x$  to position  $y$ . What is this transition probability in the case of the HMC method?

Let's see this: if we start from a state  $(x_0, p_0)$  after  $T$  steps (in a time discretization manner), we arrive at the state  $(x_T, p_T)$ . However, the Hamiltonian dynamics is deterministic, so there exist two functions  $f_1$  and  $f_2$  such that

$$x_T = f_1(x_0, p_0) \quad p_T = f_2(x_0, p_0) \quad (273)$$

Now, the reversibility of the dynamical equations tells us that if we start from the state  $(x_T, -p_T)$ , then after  $T$  steps we fall back to the initial state  $(x_0, p_0)$ . This implies that

$$x_0 = f_1(x_T, -p_T) \quad p_0 = f_2(x_T, -p_T) \quad (274)$$

This establishes a bijective correspondence between  $(x_0, p_0)$  and  $(x_T, p_T)$  on one hand, and  $(x_T, -p_T)$  on the other hand. Let's now assume that there is a unique momentum  $p^{(x,y)}$  such that between the positions  $x$  and  $y$  the following mapping holds:  $y = f_1(x, p^{(x,y)})$ , and at the same time denote  $p_y = f_2(x, p^{(x,y)})$ . Reversibility then tells us that  $x = f_1(y, -p_y)$  and  $p^{(x,y)} = f_2(y, -p_y)$ . Thus, there is only one momentum  $p^{(y,x)}$  that connects  $(x, p^{(x,y)})$  and  $(y, p^{(y,x)})$ , which is  $p^{(y,x)} = -p_y$ . Therefore,  $p_t(x \rightarrow y) = \Pi(p^{(x,y)})$  ( $\Pi(p) \propto e^{-E_c(p)}$ ) because only the momentum information moves the position value, and similarly we identify

$p_t(y \rightarrow x) = \Pi(-p_y)$ <sup>160</sup>. Now, we can prove the relation 272:

$$\begin{aligned}
\pi(x)p_t(x \rightarrow y) &= \pi(x)\Pi(p^{(x,y)}) \\
&= \frac{1}{Z}e^{-U(x)-E_c(p^{(x,y)})} = Z^{-1}e^{-H(x,p^{(x,y)})} \\
&= Z^{-1}e^{-H(y,p_y)} \quad (\text{H} = \text{Cte}) \\
&= Z^{-1}e^{-U(y)-E_c(p_y)} \\
&= \pi(y)\Pi(p_y) \\
&= \pi(y)\Pi(-p_y) \quad (E_c(p) \text{ sym. } p \leftrightarrow -p) \\
&= \pi(y)p_t(y \rightarrow x)
\end{aligned} \tag{275}$$

Having this detailed balance property, we know that  $\pi(x)$  is the invariant measure towards which the algorithm will converge.

Now, regarding the second question, let's inspect the part of the algorithm corresponding to step 2. It is a discretized form of the equations of motion, reminiscent of the Euler algorithm, for example. The most intriguing part is step 3. What is its purpose? Let's revisit the steps of one iteration in the + direction of the progression from  $(x_0, p_0)$  to a new state (the intermediate step is denoted with 1/2):

$$\begin{aligned}
\hat{p}_{1/2}^+ &= p_0^+ - \varepsilon/2 \nabla U(x_0^+) \\
\hat{x}_1^+ &= x_0^+ + \varepsilon \hat{p}_{1/2}^+ \\
\hat{p}_1^+ &= \hat{p}_{1/2}^+ - \varepsilon/2 \nabla U(x_1^+)
\end{aligned} \tag{276}$$

And finally, with step 3, we transition from  $(x_0^+, p_0^+)$  to  $(\hat{x}_1^+, -\hat{p}_1^+)$ . If we follow the same procedure but starting from  $(x_0^-, p_0^-) = (\hat{x}_1^+, -\hat{p}_1^+)$  in the opposite direction, then for the first half-step of the momentum we have:

$$\begin{aligned}
\hat{p}_{1/2}^- &= p_0^- - \varepsilon/2 \nabla U(x_0^-) \\
&= -\hat{p}_1^+ - \varepsilon/2 \nabla U(\hat{x}_1^+) \\
&= -(\hat{p}_{1/2}^+ - \varepsilon/2 \nabla U(x_1^+)) - \varepsilon/2 \nabla U(x_1^+) \\
&= -\hat{p}_{1/2}^+
\end{aligned} \tag{277}$$

---

160. I agree, there are a lot of  $p$ ,  $\pi$ , etc.

We are indeed going in the opposite direction. Regarding the position:

$$\begin{aligned}
 \hat{x}_1^- &= x_0^- + \varepsilon \hat{p}_{1/2}^- \\
 &= \hat{x}_1^+ - \varepsilon \hat{p}_{1/2}^+ \\
 &= x_0^+ + \varepsilon \hat{p}_{1/2}^+ - \varepsilon \hat{p}_{1/2}^+ \\
 &= x_0^+
 \end{aligned} \tag{278}$$

So, we return to the original position. Finally, performing the second half-step of the momentum, we have:

$$\begin{aligned}
 \hat{p}_1^- &= p_{1/2}^- - \varepsilon/2 \nabla U(x_1^-) \\
 &= -\hat{p}_{1/2}^+ - \varepsilon/2 \nabla U(x_0^+) \\
 &= -p_0^+ + \varepsilon/2 \nabla U(x_0^+) - \varepsilon/2 \nabla U(\hat{x}_0^+) \\
 &= -p_0^+
 \end{aligned} \tag{279}$$

Thus, after the step of reversing the momentum, we also recover the original value  $p_0^+$ . Therefore, the step of reversing the momentum accounts for the reversibility of the laws of Hamiltonian dynamics, which is the basis for the convergence property of the algorithm. However, in practice, since the kinetic energy expression is quadratic in  $p$ , changing the sign has no impact on the probability of accepting the new position, so we can practically remove step 3.

Now, you understand a bit better, I hope, the mechanics of the HMC algorithm. However, there are two (hyper)-parameters  $n_{steps}$  and  $\varepsilon$  whose values depend on practical considerations and experimentally influence the convergence speed of the algorithm. The notebook [https://github.com/jecampagne/cours\\_mallat\\_cdf/blob/main/2003/Monte\\_Carlo\\_Sampling.ipynb](https://github.com/jecampagne/cours_mallat_cdf/blob/main/2003/Monte_Carlo_Sampling.ipynb) provides a simple implementation in Python (`numpy`) of this algorithm. Intuitively,  $\varepsilon$  should be small, as it characterizes the discretization step of the Hamiltonian equations. However, the number of steps (`n_steps`) remains an issue: too few steps result in a random walk evolution (too much stochasticity), too many steps waste resources, and, most importantly, it can cause the chain's trajectory to loop back on itself, generating samples close to each other as if the chain were stagnant, or even causing loss of convergence properties (non-ergodicity). Therefore, determining  $n_{steps}$  is delicate and

requires preparatory simulations and a lot of practice. This difficulty made the practice of the HMC method challenging, until the development of the NUTS algorithm<sup>161</sup> in 2011.

NUTS, short for No-U-Turn Sampler, constructs a procedure that ensures the maximum distance between the position  $x_i$  and the potential new position  $x_{\text{new}}$ . We realize that if we start from  $x_i$ , the Hamiltonian dynamics gives us that the variation of  $\|x_i - x(t)\|^2$  satisfies the equation

$$C(t) = \frac{1}{2} \frac{d}{dt} \|x(t) - x_i\|^2 = (x(t) - x_i) \cdot \frac{dx(t)}{dt} = (x(t) - x_i) \cdot p(t) \quad (280)$$

Thus, we can gradually realize during the leapfrog algorithm's progression if the right-hand term becomes negative, as it would indicate a turning point in the trajectory. However, if we only simulate the part where  $C(t) > 0$ , we have an issue with reversibility. To address this problem, the NUTS algorithm introduces an auxiliary variable and constructs a binary tree of forward (direction + as used earlier) and backward (-) movements to provide movement proposals at a depth of  $2^n$ . But the complete algorithm is complex, so I refer you to the authors' article. For a user, this is where the `numpyro` library<sup>162</sup> will be useful. Note that `numpyro` is not the only library on the market for handling Markov chain generation using the HMC/NUTS method. The use of other libraries is discussed, for example, in this 2018 post: <https://mattpitkin.github.io/samplers-demo/pages/samplers-samplers-everywhere/>.

The notebook [https://github.com/jecampagne/cours\\_mallat\\_cdf/blob/main/2003/Monte\\_Carlo\\_Sampling\\_2.ipynb](https://github.com/jecampagne/cours_mallat_cdf/blob/main/2003/Monte_Carlo_Sampling_2.ipynb) demonstrates uses of NUTS as well as HMC using `numpyro`.

## 10.2 Normalizing Flows

In the course, we have seen the Score Diffusion method, which is based on a stochastic Ornstein-Uhlenbeck differential equation that can be inverted. In its forward mode,

---

161. Matthew D. Hoffman and Andrew Gelman (2011), *The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo*, arXiv:1111.4246, <https://arxiv.org/abs/1111.4246>

162. Website: <http://num.pyro.ai>; Du Phan, Neeraj Pradhan, and Martin Jankowiak, *Composable Effects for Flexible and Accelerated Probabilistic Programming in NumPyro* <https://arxiv.org/pdf/1912.11554.pdf>

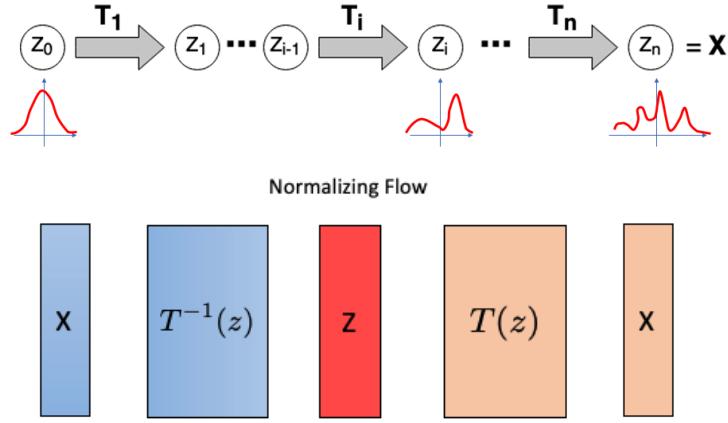


FIGURE 60 – Top: Scheme illustrating how starting from a simple distribution of  $Z_0$  and performing invertible and differentiable transformations  $T_i$ , we end up with a much more complex distribution of  $X$  (e.g., multimodal). Bottom: Scheme highlighting a two-part architecture.

we transform the random variable  $X$  following the distribution  $p(x)$ , for which we have samples  $(x_i)_i$ , progressively into a random variable  $X_T$  following the distribution  $p_T(x)$  by adding Gaussian white noise to ensure that  $p_T$  is pure noise. At the same time, we learn the score  $\nabla_x \log p_t(x)$  with noise  $\sigma_t$ . This score is used during the backward mode to generate new samples from  $p(x)$  based on samples drawn from  $p_T(x)$ , which are easy to obtain.

That being said, another technique called Normalizing Flows was used in the literature around 2018. I will not delve into a detailed study here; you can refer, for example, to the review by Papamakarios et al. (2019-21).

The general principle is relatively simple, as it involves a change of variable such that if  $Z$  has PDF  $p_z(Z)$ :

$$X = T(Z) \quad \text{with} \quad Z \sim p_z(Z) \tag{281}$$

The PDF  $p_x(X)$  of  $X$  is then obtained as follows:

$$Z = T^{-1}(X); \quad p_x(X) = p_z(Z)|\det J_T(Z)|^{-1} = p_z(T^{-1}(X))|\det J_{T^{-1}}(X)| \quad (282)$$

This is only possible if  $T$  is **invertible** and if  $T$  and  $T^{-1}$  are **differentiable functions**, as we need to compute the Jacobian  $J$  measuring the change in volume. Moreover, **this fixes the dimensionality of  $Z$  to that of  $X$** . In libraries (and in the literature), the transformation  $T$  is often called a **bijection**, with its **forward** and **backward** modes when applying  $T$  or  $T^{-1}$ , respectively.

The interest lies in the fact that we can compose transformations such that:

$$T = T_1 \circ T_2 \circ \cdots \circ T_n \quad (283)$$

with all  $T_i$  being invertible and differentiable. First, the inverse  $T^{-1}$  is easily computable as  $T^{-1} = T_n^{-1} \circ T_{n-1}^{-1} \circ \cdots \circ T_1^{-1}$ , and second, the determinant of the Jacobian of  $T$  is the product of the determinants of the Jacobians of the  $n$  transformations ( $Z_0 = Z$  and  $Z_n = X$ ):

$$Z_i = T_i(Z_{i-1}) \quad \log |\det J_T| = \sum_i \log |\det J_{T_i}(z_{i-1})| \quad (284)$$

This concept resembles the notion of probability transport discussed in the course. Thus, starting from a simple base distribution  $p_z(z)$  (e.g., a multivariate Gaussian with  $d$  variables  $\mathcal{N}(0, \mathbf{1}_d)$ ), we can model complex distributions of  $X$  as schematically shown in Figure 60. Note that computing determinants (Eq. 284) can be costly unless the transformations are special, particularly if their Jacobians are **triangular matrices**.

Given  $p_z$  and having a batch of samples  $(x_i)_{i \leq N}$ , assumed to be drawn *iid* from the  $p_x$  distribution, only the transformation  $T$  (and its components in an iterative scheme) is unknown. Suppose  $T$  belongs to a family of transformations for which we need to determine the parameters  $\phi$  to fully define it; then we proceed by *maximizing the likelihood* that the  $(x_i)_i$  are indeed drawn from the  $p_x$  distribution, i.e., by minimizing the *loss*:

$$\mathcal{L}(\phi) = - \sum_{i=1}^N \log p_x(x_i) \quad (285)$$

In fact:

- for *training*, only  $T^{-1}$  is needed, as we use the second equality of Eq. 282 to compute  $p_x(X)$  and  $\mathcal{L}(\phi)$ .
- if we want to *generate new samples* of  $X$  from  $p_x$ , it would be better to have direct access to the transformation  $T$  to use relations 281.

Obviously, we look for families of transformations that (1) allow us to compute the PDF  $p_x$  and (2) generate new samples as efficiently as possible, particularly without having to invert the transformation.

The article by Papamakarios et al. provides a series of methods for implementing these transformations, such as *autoregressive models* based on Eq. 2:

$$p(x) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdots = p(x_1) \prod_{j=2}^d p(x_j|x_1, \dots, x_{j-1}) \quad (286)$$

for modeling

$$p(x_i|x_{1:i-1}) = p(x_i; h_i) \quad h_i = C_i[x_{1:i-1}] \quad (287)$$

with neural networks.

So, the question that comes to mind is: what is the difference with the Score Diffusion method? *The main difference<sup>163</sup> with a classical normalizing flow is that the path between the data and the Gaussian is fixed; it's not an arbitrary function that we learn but it's given by the Ornstein-Uhlenbeck equation with the score. This allows having an objective for each time: we learn the path bit by bit, instead of having only a loss on the endpoint as in the maximum-likelihood training of normalizing flows.* And it is clear that while there have been generative models based on normalizing flows<sup>164</sup>, the major generative models are based on score.

---

163. Extract from a discussion with F. Guth.

164. See for example Kingma and Dhariwal (2018) <https://arxiv.org/pdf/1807.03039.pdf> (both from OpenAI).