université PARIS-SACLAY
NNT: 2016SACLS025

UNIVERSITÉ PARIS SUD
Comprendre le monde, construire l'avenir

Thèse de doctorat

de L'Université Paris-Saclay
préparée à L'Université Paris-Sud

École Doctorale n°580
Sciences et technologies de l'information et de la communication

Spécialité : Mathématiques et informatique

par
Jérémy Bensadon

Applications de la théorie de l'information à l'apprentissage statistique

Thèse présentée et soutenue à Orsay, le 2 février 2016

Composition du Jury :

M. Sylvain Arlot	Professeur, Université Paris-Sud	Président du jury
M. Aurélien Garivier	Professeur, Université Paul Sabatier	Rapporteur
M. Tobias Glasmachers	Junior Professor, Ruhr-Universität Bochum	Rapporteur
M. Yann Ollivier	Chargé de recherche, Université Paris-Sud	Directeur de thèse

Remerciements

Cette thèse est le résultat de trois ans de travail, pendant lesquels j'ai côtoyé de nombreuses personnes qui m'ont aidé, soit par leurs conseils ou leurs idées, soit simplement par leur présence, à produire ce document.

Je tiens tout d'abord à remercier mon directeur de thèse Yann Ollivier. Nos nombreuses discussions m'ont beaucoup apporté, et j'en suis toujours sorti avec les idées plus claires qu'au départ.

Je remercie également mes rapporteurs Aurélien Garivier et Tobias Glasmachers pour leur relecture attentive de ma thèse et leurs commentaires, ainsi que Sylvain Arlot, qui m'a permis de ne pas me perdre dans la littérature sur la régression et a accepté de faire partie de mon jury.

Frédéric Barbaresco a manifesté un vif intérêt pour la troisième partie de cette thèse, et a fourni de nombreuses références, parfois difficiles à trouver.

L'équipe TAO a été un environnement de travail enrichissant et agréable, par la diversité et la bonne humeur de ses membres, en particulier mes camarades doctorants. Un remerciement tout particulier aux secrétaires de l'équipe et de l'école doctorale Marie-Carol puis Olga, et Stéphanie, pour leur efficacité. Leur aide dans les différentes procédures de réinscription et de soutenance a été inestimable.

Enfin, je voudrais surtout remercier ma famille et mes amis, en particulier mes parents et mes sœurs. Pour tout.

Paris, le 27 janvier 2016
Jérémy Bensadon

Contents

Introduction

Information theory has a wide range of applications. In this thesis, we focus on two different machine learning problems, for which information theoretical insight was useful.

The first one is regression of a function $f : X \mapsto Y$. We are given a sequence (x_i) and we want to predict the $f(x_i)$ knowing the $f(x_j)$ for $j < i$. We use techniques inspired by the Minimum Description Length principle to obtain a quick and robust algorithm for online regression.

The second one is black box optimization. Black-box optimization consists in finding the minimum of a function f when the only information we have about f is a "black box" returning f. Our goal is to find the minimum of f using as few calls of the black box as possible. We transform this optimization problem into an optimization problem over a family Θ of probability distributions, and by exploiting its Riemannian manifold structure [AN07], we introduce a black-box optimization algorithm that does not depend on arbitrary user choices.

1 Regression

Density estimation and text prediction can be seen as general regression problems. Indeed, density estimation on X is regression of a random function from a singleton to X, and text prediction with alphabet A is regression of a function from $\mathbb{N}$ to A, with sample points coming in order.

This remark led us to generalize a well-known text compression algorithm, Context Tree Weighting (CTW), to general regression problems, and therefore also to density estimation. The CTW algorithm computes the Bayesian mixture of all visible Markov models for prediction.

The generalized CTW algorithm computes a similar Bayesian mixture, but on regression models using specialized submodels on partitions of D_f.

The idea of using trees for regression is not new. We give the essential characteristics of the general CTW algorithm below:

- The main motivation for the generalized CTW algorithm is the *minimum description length principle*: we want the shortest description of the data. More precisely, the prediction of $f(x_i)$ is a probability

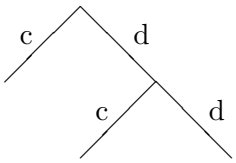


Figure 1: A Markov model for text prediction on $\{c, d\}$ (Figure 3.1). Each leaf s contains a local model for characters occuring after the context s

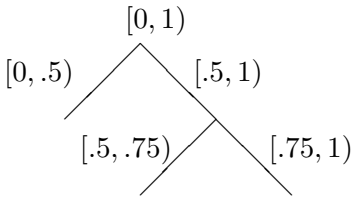


Figure 2: A Markov model for regression on $[0, 1)$. Each leaf $[a, b)$ contains a local model for regression on $[a, b)$.

distribution P on Y, and the loss is $-\ln P(f(x_i))$ (this point of view will be developed in part I). MDL-driven histogram models for density estimation have been studied in [KM07].

In several other methods where $Y = \mathbb{R}$, the prediction for $f(x_i)$ is a real number, and the loss incurred is the square of the error, this corresponds to the log loss, where the prediction is a Gaussian probability distribution around $f(x_i)$ with fixed variance.

- CTW is robust: it computes the Bayesian mixture of a very large number of models, and consequently, performs at least as well as the best of these models (up to a *constant* independent of the data). Moreover, since CTW performs model averaging and not model selection (as in [Aka09]), a small variation in the previous data cannot lead to a large variation in the predictions (unless the local models are not continuous).

- The tradeoff for this robustness is that to compute the average over all the models, the tree has to be fixed in advance (for example, by splitting the interval corresponding to a given node in two equal parts, we obtain a Bayesian mixture on all *dyadic* partition models). This is different from CART or random forests [BFOS84], [HTF01] [RG11], since the trees built by these algorithms are allowed to depend on the data.

- Contrarily to kernel methods for density estimation, or lasso and ridge regression for regression [HTF01], CTW is an online model,and conse-

quently, it has to run fast: the running time for one data point is the depth of the tree multiplied by the time complexity of the models at the nodes.

- CTW is not a model by itself, it is a way of combining models. It is close to Hierarchical Mixtures of Experts [HTF01], but the main difference with the latter is that CTW treats all nodes the same way.

2 Black-Box optimization: Gradient descents

A common approach for black-box optimization, shared for example by CMA-ES [Han11] and xNES [GSY+10], is the maintaining of a probability distribution P_t according to which new points are sampled, and updating P_t according to the sampled points at time t.

A simple way of updating P_t is a gradient descent on some space of probability distributions (we will be specifically interested in Gaussian distributions), parametrized by some Θ (e.g. mean and covariance matrix for Gaussians). We try to minimize $E_{P_t}(f)$, thus yielding the update:

$$\theta^{t+\delta t} = \theta^t - \delta t \frac{\partial E_{P_{\theta^t}}(f)}{\partial \theta} \tag{1}$$

However, there is a fundamental problem: $\frac{\partial E_{P_{\theta^t}}(f)}{\partial \theta}$ depends on the parametrization $\theta \mapsto P_\theta$.

As a simple example, suppose we are tring to minimize $f : (x, y) \mapsto x^2 + y^2$. The gradient update is $(x, y)^{t+\delta t} = (x, y)^t - \delta t(2x, 2y)$. Now, we change the parametrization by setting $x' = \frac{1}{1000}x$. We have $f(x', y) = 10^6(x')^2 + y^2$, so the new gradient update reads: $(x', y)^{t+\delta t} = (x', y)^t - \delta t(2.10^6 x', 2y)$, which corresponds to $(x, y)^{t+\delta t} = (x, y)^t - \delta t(2.10^3 x, 2y)$, a different update.

In general, for any given time step δt, by simply changing the parametrization, it is possible to reach *any* point of the half space $\frac{\partial f}{\partial \theta} < 0$, so gradient descents are not well defined.

A possible way of solving this problem is to endow Θ with a metric $\theta \mapsto I(\theta)$, and the gradient update then reads $\theta^{t+\delta t} = \theta^t - \delta t I^{-1}(\theta)\frac{\partial E_{P_{\theta^t}}(f)}{\partial \theta}$. This point of view will be developed at the beginning of part III.

The *Fisher metric* is a metric that appears naturally when considering probability distributions. The gradient associated with the Fisher metric is called the *natural gradient*, and the study of spaces of probability distributions endowed with this metric (so they are Riemannian manifolds) is called *Information Geometry* [AN07].

As shown in [OAAH11], both CMA-ES and xNES can be described as natural gradient descents, for different parametrizations. Following [OAAH11],

instead of applying a pure gradient method, we used this Riemannian manifold structure to define a fully parametrization-invariant algorithm that follows geodesics.

3 Contributions

The contributions of this thesis can be separated in two parts: the first one concerns the generalization of the Context Tree Weighting algorithm, and the second one concerns black-box optimization. They are the following:

Generalization of the Context Tree Weighting algorithm. The Context Tree Weighting algorithm efficiently computes a Bayesian combination of all visible Markov models for text compression. We generalize this idea to regression and density estimation, replacing the set of visible Markov models by (for example) the set of dyadic partitions of $[0, 1]$, which correspond to the same tree structure. The framework developed here also recovers several algorithms derived from the original CTW algorithm such as Context Tree Switching [VNHB11], Adaptive Context Tree Weighting [OHSS12], or Partition Tree Weighting [VWBG12].

Edgewise context tree algorithms. Suppose that we are regressing a function on $[0, 1)$ but we mostly saw points in $[0, 0.5)$. If the predictions on $[0, 0.5)$ are good, the posterior of the model "split $[0, 1)$ into $[0, 0.5)$ and $[0.5, 1)$ will be much greater than the posterior of the model "use the local expert on $[0, 1)$". In particular, a new point falling in $[0.5, 1)$ will be predicted with the local expert for $[0.5, 1)$, but this model will not have enough observations to issue a meaningful prediction.

The *edgewise* versions of the context tree algorithms address this by allowing models of the form "Use the local expert on $[0, 0.5)$ if the data falls there, and use the local expert on $[0, 1)$ if the data falls into $[0.5, 1)$", with the same algorithmic complexity as the non edgewise versions. The edgewise context tree weighting algorithm can even be described as a Bayesian combination similar to the regular CTW case (Theorem 4.41).

Switch behaves better than Bayes in Context Trees. We prove that in a 1-dimensional regression case, the Context Tree Switching algorithm offers better performance that the Context Tree Weighting algorithm. More precisely, for any Lipschitz regression function:

- The loss of the CTS and edgewise CTS algorithms at time t is less than $-t \ln t + O(t)$.[1] This bound holds for a larger class of sample points with the edgewise version of the algorithm (Corollaries 5.14 and 5.18).

[1] Negative losses are an artifact due to the continuous setting, with densities instead of actual probabilities. "Loss L" has to be read as "$-\ln \varepsilon + L$ bits are needed to encode the data with precision ε".

- The loss of the CTW algorithm at time t is at least $-t \ln t + \frac{1}{2} t \ln \ln t + O(t)$ (Corollary 5.20).[2]

  More generally, under the same assumptions, *any* Bayesian combination of *any* unions of experts incurs the same loss (Corollary 5.21).

A fully parametrization-invariant black-box optimization algorithm. As shown in [OAAH11], xNES [GSY+10] and CMA-ES [Han11] can be seen as natural gradient descents in $\mathbb{G}_d$ the space of Gaussian distributions in dimenstion d endowed with the Fisher metric, using different parametrizations. Although the use of the natural gradient ensures that they are equal at first order, in general, different parametrizations for $\mathbb{G}_d$ yield different algorithms. We introduce GIGO, which follows the geodesics of $\mathbb{G}_d$ and is consequently fully parametrization-invariant.

Application of Noether's Theorem to the geodesics of the space of Gaussians. Noether's theorem roughly states that if a system has symmetries, then there are invariants attached to these symmetries. It can be directly applied to computing the geodesics of $\mathbb{G}_d$. While a closed form of these geodesics has been known for more than twenty years [Eri87, CO91], Noether's theorem provides enough invariants to obtain a system of first order differential equations satisfied by the geodesics of $\mathbb{G}_d$. [3]

A description of xNES as a "Blockwise GIGO" algorithm. We show that contrary to previous intuition ([DSG+11], Figure 6), xNES does not follow the geodesics of $\mathbb{G}_d$ (Proposition 9.13). However, we can give an almost intrinsic description of xNES as "Blockwise GIGO" (Proposition 9.10): if we write $\mathbb{G}_d \cong \mathbb{R}^d \times P_d$ and split the gradient accordingly, then the xNES update follows the geodesics of these two spaces separately.

4 Thesis outline

The first part (Chapters 1 to 3) is an introduction focusing on the Minimum Description Length principle, and recalling the Context Tree Weighting [WST95] algorithm, which will be generalized in part II, and the Fisher metric, which will be used in part III. This part mostly comes from the notes of the first three talks of a series given by Yann Ollivier, the full notes can be found at `www.lri.fr/~bensadon`.

Chapter 4 introduces a formal way of handling sequential prediction, experts [4], and presents switch distributions. These tools are then used to generalize the Context Tree Weighting (CTW) and the Context Tree Switching (CTS) algorithm.

[2] See footnote 1.

[3] Theorem 8.13 gives equation (4) from [CO91], the integration constants being the Noether invariants.

[4] Experts can be seen as a generalization of Prequential Forecasting Systems [Daw84].

Chapter 5 compares the performance of the Context Tree Weighting and the Context Tree Switching algorithms on a simple regression problem. In Chapter 6, numerical experiments are conducted.

The next part led to [Ben15]. Chapter 7 presents the IGO framework [OAAH11], that can be used to describe several state of the art black-box optimization algorithms such as xNES [GSY+10] and CMA-ES [Han11][5]; and introduces Geodesic IGO, a fully parametrization-invariant algorithm. The geodesics of $\mathbb{G}_d$ the space of Gaussians in dimension d, needed for the practical implementation of the GIGO algorithm, are computed in Chapter 8.

Chapter 9 introduces two possible modifications of GIGO (twisted GIGO and blockwise GIGO) to allow it to incorporate different learning rates for the mean and the covariance.

Finally, GIGO is compared with pure rank-μ CMA-ES and xNES in Chapter 10.

[5] Actually, only pure rank-μ CMA-ES can be described in the IGO framework

Notation

We introduce here some general notation that we will use throughout this document.

- $|x|$ is the "size" of x. Its exact signification depends on the type of x (length of a string, cardinal of a set, norm of a vector...).
- If A is a set, A^* denotes the set of finite words on A.
- $\prod_{i=n}^{m} ...$ and $\sum_{i=n}^{m} ...$ when $m < n$ denote respectively an empty product, which is 1, and an empty sum, which is 0.
- We usually denote the nodes of a tree T by character strings. In the binary case, for example, the root will be ε, and the children of s are obtained by appending 0 or 1 to s. We will use prefix or suffix notation according to the context.

  We simply refer to the set of the children of a given node whenever possible.
- If $\mathbf{T}$ is a tree, we denote by $\mathbf{T}_s$ the maximal subtree of $\mathbf{T}$ with root s.
- Vectors (and strings) will be noted either with bold letters (e.g. $\mathbf{z}$) if we are not interested in their length, or with a superscript index (e.g. $z^n = (z_1, z_2, ..., z_n)$). $z^{n:m} := (z_n, z_{n+1}, ..., z_{m-1}, z_m)$ denotes the substring of z from index n to index m. If no z_0 is defined, $z^0 := \varnothing$.
- For any set X, for any $x \in X$, for any $k \in \mathbb{N} \cup \{\infty\}$, x^k is the vector (or string) of length k containing only x.
- For any set X, for $I, J \in X^{\mathbb{N}}$, we write $I \sim_n J$ if $I_i = J_i$ for $i \geqslant n$.

Part I

Information theoretic preliminaries

Chapter 1

Kolmogorov complexity

Most of the ideas discussed here can be found in [LV08].

1.1 Motivation for Kolmogorov complexity

When faced with the sequence 2 4 6 8 10 12 x, anybody would expect x to be 14. However, one could argue that the sequence 2 4 6 8 10 12 14 does not exist "more" than 2 4 6 8 10 12 13, or 2 4 6 8 10 12 0: there seems to be no reason for picking 14 instead of anything else. There is an answer to this argument: 2 4 6 8 10 12 14 is "simpler" than other sequences, in the sense that it has a shorter description.

This can be seen as a variant of Occam's razor, which states that for a given phenomenon, the simplest explanation should be preferred. This principle has been formalized in the 60s by Solomonoff [Sol64] and Kolmogorov [Kol65]. As we will soon see, the *Kolmogorov complexity* of an object is essentially the length of its shortest description, where "description of x" means "algorithm that can generate x", measured in bytes. However, in practice, finding the shortest description of an object is difficult.

Kolmogorov complexity provides a reasonable justification for "inductive reasoning", which corresponds to trying to find short descriptions for sequences of observations. The general idea is that any regularity, or structure, detected in the data can be used to compress it.

This criterion can also be used for prediction: given a sequence $x_1, ..., x_n, (?)$ choose the x_{n+1} such that the sequence $x_1, ..., x_{n+1}$ has the shortest description, or in other words, such that x_{n+1} "compresses best" with the previous x_i . For example, given the sequence 0000000?, 0 should be predicted, because 00000000 is simpler than $0000000x$ for any other x.

As a more sophisticated example, given a sequence $x_1, y_1, x_2, y_2, x_3, y_3, x_4, ...$, if we find a simple f such that $f(x_i) = y_i$, we should predict $f(x_4)$ as the next element of the sequence. With this kind of relationship, it is only nec-

essary to know the x_i, and f to be able to write the full sequence. If we also have $x_{i+1} = g(x_i)$, then only x_0,f and g have to be known: somebody who has understood how the sequence $(1, 1; 2, 4; 3, 9; 4, 16; ...)$ is made will be able to describe it very efficiently.

Any better understanding of the data can therefore be used to find structure in the data, and consequently to compress it better: comprehension and compression are essentially the same thing. This is the *Minimum Description Length* (MDL) principle [Ris78].

In the sequence $(1, 1; 2, 4; 3, 9; 4, 16; ...)$, f was very simple, but the more data we have, the more complicated f can reasonably be: if you have to learn by heart two sentences x_1, y_1, where x_1 is in English, and y_1 is x_1 in German, and you do not know German, you should just learn y_1. If you have to learn by heart a very long sequence of sentences such that x_i is a sentence in English and y_i is its translation in German, then you should learn German. In other words, the more data we have, the more interesting it is to find regularity in them.

The identity between comprehension and compression is probably even clearer when we consider text encoding: with a naive encoding, a simple text in English is around 5 bits for each character (26 letters, space, dot), whereas the best compression algorithms manage around 3 bits per character. However, by removing random letters from a text and having people try to read it, the actual information has been estimated at around 1 bit per character ([Sha51] estimates the entropy between .6 and 1.3 bit per character).

1.2 Formal Definition

Let us now define the Kolmogorov complexity formally:

Definition 1.1. *The Kolmogorov complexity of x a sequence of 0s and 1s is by definition the length of the shortest program on a Universal Turing machine*[1] *that prints x. It is measured in bits.*[2]

The two propositions below must be seen as a sanity check for our definition of Kolmogorov complexity. Their proofs can be found in [LV08].

Proposition 1.2 (Kolmogorov complexity is well-defined)**.** *The Kolmogorov complexity of x does not depend on the Turing machine, up to a constant which does* not *depend on x (but does depend on the two Turing machines).*

Sketch of the proof. if P_1 prints x for the Turing machine T_1, then if I_{12} is an interpreter for language 1 in language 2, $I_{12} :: P_1$ prints x for the Turing machine T_2, and therefore $K_2(x) \leqslant K_1(x) + \text{length}(I_{12})$ □

[1] Your favorite programming language, for example.

[2] Or any multiple of bits.

In other words, if P is the shortest zipped program that prints x in your favourite programming language, you can think about the Kolmogorov complexity of x as the size (as a file on your computer) of P (we compress the program to reduce differences from alphabet and syntax between programming languages).

Since Kolmogorov complexity is defined up to an additive constant, all inequalities concerning Kolmogorov complexity are only true up to an additive constant. Consequently, we write $K(x) \overset{+}{\leqslant} f(x)$ for $K(x) \leqslant f(x) + a$, where a does not depend on x:

Notation 1.3. *Let X be a set, and let f, g be two applications from X to $\mathbb{R}$.*

We write $f \overset{+}{\leqslant} g$ if there exists $a \in \mathbb{R}$ such that for all $x \in X$, $f(x) \leqslant g(x) + a$.

If the objects we are interested in are not sequences of 0 and 1 (pictures, for example), they have to be encoded.

Proposition 1.4. *the Kolmogorov complexity of x does not depend on the encoding of x, up to a constant which does* not *depend on x (but does depend on the two encodings).*

Sketch of the proof. Let f,g be two encodings of x. We have $K(g(x)) < K(f(x)) + K(g \circ f^{-1})$ (instead of encoding $g(x)$, encode $f(x)$, and the map $g \circ f^{-1}$. $\max\left(K(g \circ f^{-1}), K(f \circ g^{-1})\right)$ is the constant). □

In other words, the Kolmogorov complexity of a picture will not change if you decide to put the most significant bytes for your pixels on the right instead of the left.

Notice that the constants for these two propositions are usually reasonably small (the order of magnitude should not exceed the megabyte, while it is possible to work with gigabytes of data).

1.3 Kolmogorov complexity is not computable

Kolmogorov complexity is not computable. Even worse, it is never possible to prove that the Kolmogorov complexity of an object is large.

An intuitive reason for the former fact is that to find the Kolmogorov complexity of x, we should run all possible programs in parallel, and choose the shortest program that outputs x, but we do not know when we have to stop: There may be a short program still running that will eventually output x. In other words, it is possible to have a program of length $K(x)$ that outputs x, but it is *not* possible to be sure that it is the shortest one.

Theorem 1.5 (Chaitin's incompleteness theorem)**.** *There exists a constant* L[3] *such that it is not possible to prove the statement* $K(x) > L$ *for any* x*.*

Sketch of the proof. For some L, write a program that tries to prove a statement of the form $K(x) > L$ (by enumerating all possible proofs). When a proof of $K(x) > L$ for some x is found, print x and stop.

If there exists x such that a proof of $K(x) > L$ exists, then the program will stop and print some x_0, but if L has been chosen largen enough, the length of the program is less than L, and describes x_0. Therefore, $K(x_0) \leqslant L$. contradiction. □

This theorem is proved in [Cha71] and can be linked to Berry's paradox:
"The smallest number that cannot be described in less than 13 words"
"The [first x found] that cannot be described in less than [L] bits".

Corollary 1.6. *Kolmogorov complexity is not computable.*

Proof. Between 1 and 2^{L+1}, there must be at least one integer n_0 with Kolmogorov complexity greater that L (since there are only $2^{L+1} - 1$ programs of length L or less). If there was a program that could output the Kolmogorov complexity of its input, we could prove that $K(n_0) > L$, which contradicts Chaitin's theorem.

□

As a possible solution to this problem, we could define $K_t(x)$, the length of the smallest program that outputs x in less than t instructions, but we lose some theoretical properties of Kolmogorov complexity (Proposition 1.2 and 1.4 have to be adapted to limited time but they are weaker, see [LV08], Section 7. For example, Proposition 1.2 becomes $K_{ct \log_2 t,1}(x) \leqslant K_{t,2}(x)+c$).

[3] reasonably small, around 1Mb

Chapter 2

From Kolmogorov Complexity to Machine Learning

Because of Chaitin's theorem, the best we can do is finding upper bounds on Kolmogorov complexity. First, we introduce *prefix-free* Kolmogorov complexity and prove *Kraft's inequality*, and state Shannon encoding theorem. Finally, we give classical upper bounds on Kolmogorov complexity, firstly for integers, then for other strings.

2.1 Prefix-free Complexity, Kraft's Inequality

If we simply decide to encode an integer by its binary decomposition, then, we do not know for example if the string "10" is the code for 2, or the code for 1 followed by the code for 0.

Similarly, given a Turing machine, and two programs P_1 and P_2, there is nothing that prevents the concatenation P_1P_2 from defining another program that might have nothing to do with P_1 or P_2.

This leads us to the following (not formal) definition:

Definition 2.1. *A set of strings S is said to be* prefix-free *is no element of S is a prefix of another.*

A code is said to be prefix-free *if no codeword is a prefix of another (or equivalently, if the set of codewords is prefix-free).*

We then adapt the definition of Kolmogorov complexity by forcing the set of programs to be prefix-free (hence the name *prefix-free* Kolmogorov complexity[1]).

[1] "self-delimited" is sometimes used instead of prefix-free.

It is clear now that if we receive a message encoded with a prefix-free code, we can decode it unambiguously letter by letter, while we are reading it, and if a Turing machine is given two programs P_1 and P_2, their concatenation will not be ambiguous.

With programming languages, for example, working with prefix-free complexity does not change anything, since the set of compiling programs is prefix free (the compiler is able to stop when the program is finished).

However, being prefix-free is a strong constraint: a short codeword forbids many longer words. More accurately, we have the following result:

Proposition 2.2 (Kraft's inequality)**.** *Let $\mathcal{S} \subset \{0,1\}^*$ be the set of prefix-free set of strings, and let $S \in \mathcal{S}$. We have*

$$\sum_{s \in S} 2^{-|s|} \leqslant 1 \tag{2.1}$$

Sketch of the proof. We construct an application ϕ from $\mathcal{S}$ to the set of binary trees: for $S \in \mathcal{S}$, $\phi(S)$ is the smallest full binary tree T such that all elements of s are leaves of T. Such a tree always exists: start with a deep enough tree[2], and close nodes until each pair of "sibling leaves" contain either an element of S or a node that has at least one element of s as its children. No internal node can correspond to an element of S, because it would then be the prefix of another element of s.

So finally, we have:

$$\sum_{s \in S} 2^{-|s|} \leqslant \sum_{s \text{ leaves of } \phi(S)} 2^{-|s|} = 1 \tag{2.2}$$

□

For the general proof, see Theorem 1.11.1 in [LV08].

2.2 Classical lower bounds for Kolmogorov complexity

2.2.1 Coding integers

Now, let us go back to integers:

Proposition 2.3. *Let n be an integer. We have*

$$K(n) \overset{+}{\leqslant} \log_2 n + 2 \log_2 \log_2 n. \tag{2.3}$$

[2] We overlook the "infinite S" case, for example, $S = \{0, 10, 110, 1110, ...\}$, but the idea behind a more rigorous proof would be the same.

Proof, and a bit further. Let n be an integer, and let us denote by b its binary expansion, and by l the length of its binary expansion (i.e. $l = \lfloor \log_2(n+1) \rfloor \sim \log_2 n$).

Consider the following prefix codes (if c is a code, we will denote by $c(n)$ the codeword used for n):

- c_1: Encode n as a sequence of n ones, followed by zero. Complexity n.
- c_2: Encode n as $c_1(l) :: b$, with l and b as above. To decode, simply count the number k of ones before the first zero. The k bits following it are the binary expansion of n. Complexity $2\log_2 n$.
- c_3: Encode n as $c_2(l) :: b$. To decode, count the number k_1 of ones before the first zero, the k_2 bits following it are the binary expansion of l, and the l bits after these are the binary expansion of n. Complexity $\log_2 n + 2\log_2 \log_2 n$, which is what we wanted.
- We can define a prefix-free code c_i by setting $c_i(n) = c_{i-1}(l) :: b$. The complexity improves, but we get a constant term for small integers, and anyway, all the corresponding bounds are still equivalent to $\log_2 n$ as $n \to \infty$.
- It is easy to see that the codes above satisfy $c_n(1) = n + 1$: for small integers, it would be better to stop the encoding once a number of length one (i.e. one or zero) is written. Formally, this can be done the following way: consider $c_{\infty,1}$ defined recursively as follows :

$$c_{\infty,1}(n) = c_{\infty,1}(l-1) :: b :: 0, \tag{2.4}$$
$$c_{\infty,1}(1) = 0. \tag{2.5}$$

  It is easy to see that $c_{\infty,1}(n)$ begins with 0 for all n, i.e, $c_{\infty,1} = 0 :: c_{\infty,2}$.

  We can now set $c_\infty(0) = 0$ and $c_\infty(n) = c_{\infty,2}(n+1)$.

The codes c_2 and c_3 are similar to Elias gamma and delta (respectively) coding. c_∞ is called Elias omega coding [Eli75]. □

2.2.2 Generic bounds

We also have the following bounds:

1. A simple program that prints x is simply print(x). The length of this program is the length of the function print, plus the length of x, but it is also necessary to provide the length of x to know when the print function stops reading its input (because we are working with prefix-free Kolmogorov complexity). Consequently

$$K(x) \overset{+}{\leqslant} |x| + K(|x|). \tag{2.6}$$

By counting arguments, some strings x have a Kolmogorov complexity larger than $|x|$. These strings are called *random* strings by Kolmogorov. The justification for this terminology is that a string that cannot be compressed is a string with no regularities.

2. The Kolmogorov complexity of x is the length of x compressed with the *best* compressor for x. Consequently, we can try to approximate it with any standard compressor, like zip, and we have:

$$K(x) \overset{+}{\leqslant} |\text{zip}(x)| + |\text{unzip\_program}|. \tag{2.7}$$

This property has been used to define the following distance between two objects: $d(x,y) = \dfrac{\max\left(K(x)|y), K(y|x)\right)}{\max\left(K(x), K(y)\right)}$. By using distance-based clustering algorithms, the authors of [CV] have been able to cluster data (MIDI files, texts...) almost as anyone would expected (the MIDI files were clustered together, with subclusters essentially corresponding to the composer, for example). In the same article, the Universal Declaration of Human Rights in different languages has been used to build a satisfying language tree.

3. If we have some finite set E such that $x \in E$, then we can simply enumerate all the elements of E. In that case, an x can be described as "the n^{th} element of E. For this, we need $K(E)$ bits to describe E, and $\lceil \log_2 |E| \rceil$ bits to identify x in E:

$$K(x) \overset{+}{\leqslant} K(E) + \lceil \log_2 |E| \rceil. \tag{2.8}$$

4. More generally,

Theorem 2.4. *If μ is a probability distribution on a set X, and $x \in X$, we have*

$$K(x) \overset{+}{\leqslant} K(\mu) - \log_2(\mu(x)). \tag{2.9}$$

For example, if μ is uniform, we find equation (2.8). Another simple case is the i.i.d. case, which will be discussed later. This inequality is the most important one for machine learning, because a probability distribution μ such that $K(\mu) - \log_2(\mu(x))$ is small can be thought of as a "good" description of x, and we can expect it to predict upcoming terms well. The proof of this inequality consists in noting that coding x with $-\log_2 \mu(x)$ bits satisfies Kraft's inequality.

2.3 Probability distributions and coding: Shannon encoding theorem

Let X be a countable alphabet, and suppose we are trying to encode efficiently a random sequence of characters $(x_i) \in X^{\mathbb{N}}$, such that $P(x_n|x^{n-1})$ is known:

For all n, we are trying to find the codelength L minimizing

$$E(L(x^n)) = \sum_{x^n \in A^n} P(x^n) L(x^n), \tag{2.10}$$

under the constraint given by Kraft's inequality:

$$\sum_{x^n \in A^n} 2^{-L(x^n)} \leqslant 1 \tag{2.11}$$

The solution of this optimization problem is $L(x) := -\log_2 P(x)$, and from now on, we will identify codelengths functions L with probability distributions 2^{-L}.

Definition 2.5. *Let X be a countable alphabet, let μ and ν be probability distributions on X, and let (x_i) be a sequence of i.i.d random variables following μ.*

The quantity $H(\mu) = -\sum_{x \in X} \mu(x) \log_2 \mu(x)$ is called entropy *of μ. It is the expected number of bits needed to encode an x sampled according to μ when using μ.*

The quantity $\mathrm{KL}(\mu \| \nu) := \sum_{x \in X} \nu(x) \log_2 \frac{\nu(x)}{\mu(x)}$ is called Kullback-Leibler divergence *from ν to μ. It is the expected additional number of bits needed to encode if x is sampled according to ν, and we are using μ instead of ν.*

The Kullback–Leibler divergence is always positive. In particular, the entropy of a probability distribution μ is the minimal expected number of bits to encode an x sampled according to μ.

Proposition 2.6. *The Kullback–Leibler divergence is positive:*

Let X be a countable alphabet, and let μ, ν be two probability distributions on X. We have:

$$\mathrm{KL}(\mu \| \nu) \geqslant 0 \tag{2.12}$$

Proof.

$$\mathrm{KL}(\mu \| \nu) := \sum_{x \in X} \nu(x) \log_2 \frac{\nu(x)}{\mu(x)} \tag{2.13}$$

$$\geqslant \log_2 \left(\sum_{x \in X} \nu(x) \frac{\nu(x)}{\mu(x)} \right) = 0 \tag{2.14}$$

□

In all this section, we have been suggesting to encode any x with $-\log_2 \mu(x)$ bits for some μ, without checking that $-\log_2 \mu(x)$ is an integer. We now justify this apparent oversight.

2.3.1 Non integer codelengths do not matter: Arithmetic coding

The idea behind (2.9) is that for a given set E, if I think some elements are more likely to appear than others, they should be encoded with fewer bits. For example, if in the set $\{A, B, C, D\}$, we have $P(A) = 0.5$, $P(B) = 0.25$, and $P(C) = P(D) = 0.125$, instead of using a uniform code (two bits for each character), it is better to encode for example[3] A with 1, B with 01, C with 001 and D with 000.

In the first example, the expected length with the uniform code is 2 bits per character, while it is 1.75 bits per character for the other.

In general, it can be checked that the expected length is minimal if the length of the code word for x is $-\log_2(\mu(x))$. If we have a code satisfying this property, then (2.9) follows immediately (encode μ, and then use the code corresponding to μ to encode x).

However, if we stick to encoding one symbol after another approximations have to be made, because we can only have integer codelengths. For example, consider we have: $P(A) = 0.4$, $P(B) = P(C) = P(D) = P(E) = 0.15$. The $-\log_2 P(*)$ are not integers: we have to assign close integer codelengths. We describe two possible ways of doing this:

- Sort all symbols by descending frequency, cut when the cumulative frequency is closest to 0.5. The codes for the symbols on the left (resp. right) start with 0 (resp. 1). Repeat until all symbols have been separated. This is Shannon–Fano coding ([Fan61], similar to the code introduced in the end of the proof of Theorem 9 in [Sha48]).

- Build a binary tree the following way: Start with leave nodes corresponding to the symbols, with a weight equal to their probability. Then, take the two nodes without parents with lowest weight, and create a parent node for them, and assign it the sum of its children's wieght. Repeat until only one node remains. Then, code each symbol with the sequence of moves from the root to the leaf (0 corresponds to taking the left child, for example). This is Huffman coding [Huf52], which is better than Shannon–Fano coding.

On the example above, we can find the following codes (notice that some conventions are needed to obtain well-defined algorithms from what is described above: for Shannon-Fano, what to do when there are two possible cuts, and for Huffman, which node is the left child and which node is the right child):

[3] We do not care about the code, we care about the length of the code words for the different elements of X.

	Theoretical optimal length	Shannon–Fano code	Huffman code
A	≈ 1.322	00	0
B	≈ 2.737	01	100
C	≈ 2.737	10	101
D	≈ 2.737	110	110
E	≈ 2.737	111	111
Length expectation	≈ 2.17	2.3	2.2

As we can see, neither Shannon–Fano coding nor Huffman coding reach the optimal bound.

However, if instead of encoding each symbol separately, we encode the whole message (2.9) can actually be achieved up to a constant number of bits for *the whole message*[4] by describing a simplification of *arithmetic coding* [Ris76]:

The idea behind arithmetic coding is to encode the whole message as a number in the interval $[0, 1]$, determined as follow: consider we have the message $(x_1, ..., x_N) \in X^N$ (here, to make things easier, we fix N). We start with the full interval, and we partition it in $\#X$ subintervals, each one corresponding to some $x \in X$ and of length our expected probability to see x, given the characters we have already seen, and we repeat until the whole message is read. We are left with an subinterval I_N of $[0, 1]$, and we can send the binary expansion of any number in I_N (so we choose the shortest one).

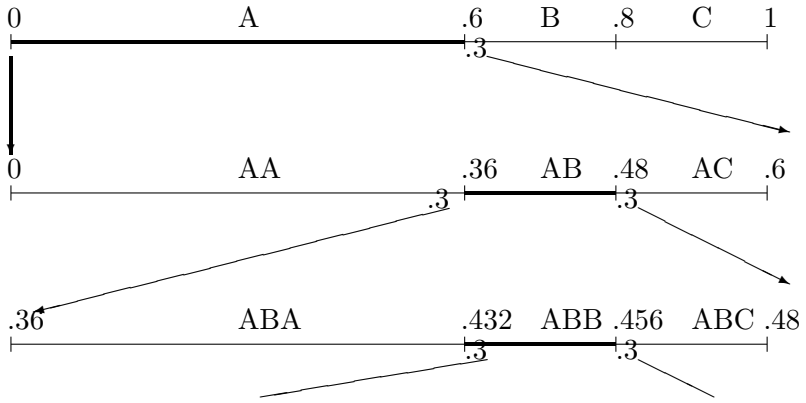


Figure 2.1: Arithmetic coding of a word starting with ABB, with $P(A) = 0.6$, $P(B) = 0.2$, $P(C) = 0.2$

[4]Since all the inequalities were already up to an additive constant, this does not matter at all.

Algorithm 2.7 (End of the proof of (2.9): A simplification of arithmetic coding)**.** *We are given an ordered set X, and for x_1,...,x_n,$y \in X$, we denote by $\mu(y|x_1, ..., x_n)$ our expected probability to see y after having observed $x_1, ..., x_n$.*[5]

Goal: *Encode the whole message in one number $0 \leqslant x \leqslant 1$.*

Encoding algorithm

Part 1*: Encode the message as an interval.*

$i = 1$
$I = [0, 1]$
while $x_{i+1} \neq$ END ***do***
Partition I into subintervals $(I_x)_{x \in X}$ such that:
$x < y \implies I_x < I_y$,[6] $\text{length}(I_x) = \text{length}(I)\mu(x|x_1, ..., x_{i-1})$
Observe x_i
$I = I_{x_i}$
$i = i + 1$
end while
return I

Part 2*: pick a number in the interval I.*

We can find a binary representation for any real number $x \in [0, 1]$, by writing $x = \sum_1^{+\infty} \frac{a_i}{2^i}$, with $a_i \in \{0, 1\}$. Now, for a given interval I, pick the number $x_I \in I$ which has the shortest binary representation.[7] *The message will be encoded as x_I.*

Decoding algorithm*. x_I received.*

$i = 1$
$I = [0, 1]$

[5] A simple particular case is the case where μ does not depend on past observations (i.e. $\mu(y|x_1, ..., x_n) =: \mu(y)$) and can therefore be identified with a probability distribution on X

[6] If I, J are intervals, we write $I < J$ for $\forall x \in I$, $\forall y \in J$, $x < y$.

[7] We call *length of the representation* $(a_1, ..., a_n, ...)$ the number $\min\{n \in \mathbb{N}, \forall k > n, a_k = 0\}$. If $\text{length}(I) \neq 0$, I necessarily contains number with a finite representation.

while *not termination criterion*[8] ***do***
  Partition I *into subintervals* $(I_x)_{x\in X}$ *as in the encoding algorithm.*
  $x_i \leftarrow$ *the only* y *such that* $x_I \in I_y$
  $I = I_{x_i}$
  $i = i + 1$
end while

Arithmetic coding allows to find a code such that the length of the code word for $x = (x_1, ..., x_n)$ is $\sum_{i=1}^{n} -\log_2(\mu(x_i|x_1, ..., x_{i-1})) = \sum_{i=1}^{n} -\log_2(\mu(x_i|x_1, ..., x_{i-1})$, which is what we wanted.

However, arithmetic cannot be implemented like this, because of problems of finite precision: if the message is too long, the intervals of the partition might become undistinguishable. It is possible to give an online version of this algorithm which solves this problem: the encoder sends a bit once he knows it (he can send the n-th bit if I contains no multiple of 2^{-n}). In that case, he does not have to worry about the n first digits of the bounds of the intervals of the partition anymore, which solves the problem of finite precision. In the case 0.5 remains in the interval, the encoder can remember that if the first bit is 0, then the second is 1, and conversely, thus working in $[.25, .75]$ instead of $[0, 1]$ (and the process can be repeated).

Arithmetic coding ensures that we can focus on codelengths, instead of actual codes.

2.4 Model selection and Kolmogorov complexity

The term $-\log_2(\mu(x))$ in Theorem 2.4 is essentially the cost of encoding the data with the help of the model, whereas $K(\mu)$ can be seen as a penalty for complicated models (which, in machine learning, prevents overfitting: if the data is "more compressed", including the cost of the model and the decoder, the description is better).

As a basic example, if μ is the Dirac distribution at x, $K(\mu) = K(x)$: in

[8] There are several possibilities for the termination criterion:

- We can have an END symbol, to mark the end of the message (in that case, stop when END has been decoded).
- We can give the length N of the message first (in that case, stop when N characters have been decoded)
- The message sent by the encoder can be slightly modified as follows. We decide that a sequence of bits $(a_1, ..., a_n)$ represents the interval I_a of all numbers of which the binary expansion starts with $(a_1, ..., a_n)$. Now, the message a sent by the encoder is the shortest non ambiguous message, in the sense that I_a is contained in I, but is not contained in any of the subintervals I_x corresponding to I (where I is the interval obtained at the end of the first part of the encoding algorithm).

that case, all the complexiy is in the model, and the encoding of the data is completely free.

More interestingly, if we are trying to fit the data $x_1, ..., x_n$ to an i.i.d. Gaussian model (which corresponds to the description: "this is Gaussian noise"), with mean m and fixed variance σ^2, the term $-\log_2(\mu(x))$ is equal to $\sum_i (x_i - m)^2$ up to additive constants, and the m we should select is the solution to this least square problem, which happens to be the sample mean (if we neglect $K(m)$, which corresponds to finding the maximum likelihood estimate).

In general, $K(\mu)$ is difficult to evaluate. There exists two classical approximation, yielding different results:

- $K(\mu)$ can be approximated by the number of parameters in μ. This gives the Akaike Information Criterion (AIC) [Aka73].
- $K(\mu)$ can also be approximated by half the number of parameters in μ multiplied by the logarithm of the number of observations. This gives the Bayesian Information Criterion (BIC) [Sch78]. The reason for this approximation will be given in Section 3.1.3.

2.5 Possible approximations of Kolmogorov complexity

Now, even with these upper bounds, in practice, it is difficult to find good programs for the data.

As an introduction to the next chapter, we give some heuristics:

- Usual compression techniques (like zip) can yield good results (as in [CV], for example).
- Minimum description length techniques ([Ris78], [Gru07] for example): starting from naive generative models to obtain more complex ones: for example, if we have to predict sequences of 0 and 1, and we initially have two experts, one always predicting 0 and the other predicting always 1, we can use a mixture of experts: use the first experts with probability p, and the second with probability $1-p$, and we can obtain all Bernoulli distributions. More interestingly, we can also automatically obtain strategies of the form "after having observed x_i, use expert k to predict x_{i+1}", thus obtaining Markov models.
- Auto-encoders can also be used: they are hourglass shaped neural networks (fewer nodes in the intermediate layer), trained to output exactly the input. In that case, the intermediate layer is a compressed form of the data, and the encoder and decoder are given by the network.

- The model class of Turing machine is very large. For example, if we restrict ourselves to finite automata, we can compute the restricted Kolmogorov complexity, and if we restrict ourselves to visible Markov models we can even use Kolmogorov complexity for prediction.

Chapter 3

Universal probability distributions

The equivalence between coding and probability distribution, combined with Occam's razor is probably the main reason for defining Kolmogorov complexity: Kolmogorov complexity gives the length of the "best" coding (according to Occam's razor), so the probability distribution it defines must be the "best" predictor or generative model.

More precisely, we can define the following probability distributions on X^* (All finite sequences of elements of X. These distributions are defined up to a normalization constant for programs that do not end), that should be usable for any problem [Sol64].

$$P_1(x) = 2^{-K(x)}, \tag{3.1}$$

$$P_2(x) = \sum_{\text{all deterministic programs}} 2^{-|p|} \mathbb{1}_{p \text{ outputs } x}, \tag{3.2}$$

$$P_3(x) = \sum_{\text{all random programs}} 2^{-|p|} P(p \text{ outputs } x), \tag{3.3}$$

$$P_4(x) = \sum_{\text{probability distributions}} 2^{-|\mu|} \mu(x), \tag{3.4}$$

where a program is any string of bits to be read by a universal Turing machine, and a random program is a program that has access to a stream of random bits (in particular, a deterministic program is a random program), and $|\mu|$ is the Kolmogorov complexity of μ, i.e., the length of the shortest program computing μ (if μ is not computable, then its Kolmogorov complexity is infinite, and μ does not contribute to P_4: the sum is in practice restricted to computable probability distributions). For example, P_2 is the output of a program written at random (the bits composing the program are random, but the program is deterministic), and P_3 is the output of a random program written at random.

Notice that $P_1(x)$ can be rewritten as $\sum_{\text{Diracs}} 2^{-|\delta|}\delta(x)$: P_4 is to P_1 what P_3 is to P_2 (the Diracs are the "deterministic probability distributions").

Since Kolmogorov complexity is defined only up to an additive constant, the probability distributions above are only defined up to a multiplicative constant. This leads us to the following definition:

Definition 3.1. *Let P and P' be two probability distributions on a set X. P and P' are said to be* equivalent *if there exists two constants m and M such that:*

$$mP \leqslant P' \leqslant MP \tag{3.5}$$

Proposition 3.2 (proved in [ZL70])**.** *P_1, P_2, P_3 and P_4 are equivalent.*

Consequently, we can pick any of these probability distributions (which are called Solomonoff universal prior) as a predictor. We choose P_4:

$$P_4(x_{k+1}|x_k, ...x_1) := \frac{\sum_\mu 2^{-|\mu|}\mu(x_1, ..., x_{k+1})}{\sum_{\mu,x'} 2^{-|\mu|}\mu(x_1, ..., x_k, x')} \tag{3.6}$$

$$= \frac{\sum_\mu 2^{-|\mu|}\mu(x_1, ..., x_k)\mu(x_{k+1}|x_1, ..., x_k)}{\sum_\mu 2^{-|\mu|}\mu(x_1, ..., x_k)} \tag{3.7}$$

$$= \frac{\sum_\mu w_\mu \mu(x_{k+1}|x_1, ..., x_k)}{\sum_\mu w_\mu}, \tag{3.8}$$

where $w_\mu = 2^{-|\mu|}\mu(x_1, ..., x_k)$ can be seen as a Bayseian posterior on the probability distributions (we had the prior $2^{-|\mu|}$). In particular, *the posterior weights depend on the data.*

However, as we have seen, the Kolmogorov complexity and therefore P_4 are not computable.

We have to replace Kolmogorov complexity by something simpler: we choose a family of probability distibutions $\mathcal{F}$, and restrict ourselves to this family. The distribution we obtain is therefore $P_\mathcal{F} := \sum_{\mu\in\mathcal{F}} 2^{-|\mu|}\mu$.

We now give some possible families:

- $\mathcal{F} = \{\mu\}$. One simple model that explains past data can be good enough, and the computation cost is small (depending on μ).
- $\mathcal{F} = \{\mu_1, ..., \mu_n\}$: as seen in equation (3.8), we obtain a Bayesian combination.
- $\mathcal{F} = \{\mu_\theta, \theta \in \Theta\}$: this case will be studied below. As a simple example, we can take $\Theta = [0, 1]$, and μ_θ is Bernoulli, with parameter θ.

In that case, we have

$$P_{\mathcal{F}} = \sum_{\theta \in \Theta} 2^{-K(\theta)} \mu_\theta. \tag{3.9}$$

Notice that the right side of equation (3.9) is only a countable sum: if θ is not computable, then $K(\theta) = +\infty$, and the corresponding term does not contribute to the sum.

There exist different techniques for approximating $P_{\mathcal{F}}$ in the parametric case (see for example [Gru07]):

1. Encoding the best θ_0 to encode the data. In that case, $\sum_{\theta \in \Theta} 2^{-K(\theta)} \mu_\theta$ is approximated by the largest term of the sum. Since we encode first a $\theta \in \Theta$, and then the data, with the probability distribution P_θ, these codes are called *two-part codes*.

2. We can replace the penalization for a complex θ by a continuous Bayesian prior q on Θ. In that case, $\sum_{\theta \in \Theta} 2^{-K(\theta)} \mu_\theta$ is approximated by

$$\int_\Theta q(\theta) \mu_\theta. \tag{3.10}$$

   As we will see later, there exists a prior (called Jeffreys' prior) with good theoretical properties with respect to this construction.

3. Normalized maximum likelihood techniques (will not be discussed here)

4. We can make online predictions in the following way: use a default prior for x_1, and to predict (or encode) x_k, we use past data to choose the best μ_θ.[1] With this method, the parameter θ is defined implicitly in the data: there is no need to use bits to describe it.

We also introduce the definition of *prequential* models [Daw84]. Although we will not use it directly, it will be an important inspiration for the definition of "experts" in Section 4.1:

Definition 3.3. *A generative model P is said to be* prequential *(contraction of predictive-sequential, see [Gru07]) if* $\sum_x P(x_1, ..., x_k, x) = P(x_1, ..., x_k)$ *(i.e. different time horizons are compatible).*

A predictive model Q is said to be prequential if the equivalent generative model is prequential, or equivalently, if $Q(x_{k+1}, x_k | x_1, ..., x_{k-1}) = Q(x_{k+1} | x_1, ..., x_k) Q(x_k | x_1, ..., x_{k-1})$ (i.e. predicting two symbols simultaneously and predicting them one after the other are the same thing, hence the name "predictive-sequential")

[1] Here, "the best" does not necessarily mean the maximum likelihood estimator: if a symbol does not appear, then it will be predicted with probability 0, and we do not want this. A possible solution is to add fictional points of data before the message, which will also yield the prior on x_0. For example, when encoding the results of a game of heads or tails, it is possible to add before the first point a head, and a tail, each with weight 1/2.

It can be checked that the Bayesian model and the online model are prequential, whereas the two others are not.

Notice that prequential models really correspond to one probability distribution on X^n. The reason why two-part codes and NML codes are not prequential is that for any fixed k, they correspond to a probability distriution P_k on the set of sequences of symbols of length k, but for $k' \neq k$, P_k and $P_{k'}$ are not necessarily related.

We now study the first of these techniques: two-part codes.

3.1 Two-part codes

Our strategy is to use the best code θ_0 in our family (P_θ). Since the decoder cannot know which P_θ we are using to encode our data, we need to send θ_0, first.

Since θ_0 is a real parameter, we would almost surely need an infinite number of bits to encode it exactly: we will encode some θ "close" to θ_0 instead.

Suppose for example that $\theta_0 \in [0,1]$. If we use only its k first binary digits for θ, then we have $|\theta - \theta_0| \leqslant 2^{-k}$.

Consequently, let us define the precision of the encoding of some $\theta \in [0,1]$ with $\varepsilon := 2^{-\text{number of bits to encode } \theta}$ (so we immediately have the bound $|\theta - \theta_0| \leqslant \varepsilon$).

We recall the bound (2.9): for any probability distribution μ,

$$K(x) \leqslant K(\mu) - \log_2(\mu(x)), \tag{3.11}$$

which corresponds to coding μ, and then, using an optimal code with respect to μ to encode the data.

Here, increasing the precision (or equivalently, reducing ε) increases the likelihood of the data (and consequently $-\log_2(\mu(x))$ decreases), but $K(\mu)$ increases: we can suppose that there exists some optimal precision ε^*. Let us compute it.

3.1.1 Optimal precision

In the ideal case (infinite precision), we encode the data $x_1, ..., x_n$ by sending $\theta^* := \text{argmax}_\theta \mu_\theta(x_1, ..., x_k)$, and then, the data encoded with the probability distribution μ_{θ^*}.

The codelength corresponding to a given ε is:

$$l(\varepsilon) := -\log_2 \varepsilon - \log_2(\mu_\theta(x)), \tag{3.12}$$

where $|\theta - \theta^*| \leqslant \varepsilon$ (i.e. we encode θ, which takes $-\log_2 \varepsilon$ bits, and then, we encode x using μ_θ, which takes $-\log_2(\mu_\theta(x))$ bits).

With a second order Taylor expansion around θ^*, we find:

$$l(\varepsilon) = -\log_2 \varepsilon - \log_2 \mu_{\theta^*}(x) + \frac{\partial(-\log_2 \mu_\theta(x))}{\partial\theta}(\theta-\theta^*) + \frac{\partial^2}{\partial\theta^2}(-\log_2 \mu_\theta(x))\frac{(\theta-\theta^*)^2}{2} + o((\theta-\theta^*)^2), \tag{3.13}$$

where the derivatives are taken at $\theta = \theta^*$.

The first order term is equal to zero, since by definition, μ_{θ^*} is the probability distribution minimizing $-\log_2 \mu_\theta(x)$. If we approximate $\theta - \theta^*$ by ε and write $J(\theta^*) := \frac{\partial^2}{\partial\theta^2}(-\ln \mu_\theta(x))$ (which is positive), we find:

$$l(\varepsilon) \approx -\log_2(\mu_{\theta^*}(x)) - \log_2 \varepsilon + \frac{J(\theta^*)}{\ln 2}\frac{\varepsilon^2}{2}. \tag{3.14}$$

Differentiating with respect to ε, we find $\frac{\mathrm{d}l}{\mathrm{d}\varepsilon} = \frac{1}{\ln 2}\left(-\frac{1}{\varepsilon} + \varepsilon J(\theta^*)\right)$. If we denote by ε^* the optimal precision, we must have $\frac{\mathrm{d}l}{\mathrm{d}\varepsilon}|_{\varepsilon=\varepsilon^*} = 0$, i.e.

$$\varepsilon^* \approx \sqrt{\frac{1}{J(\theta^*)}}, \tag{3.15}$$

which, by plugging (3.15) into (3.14), yields the following codelength:

$$l(\varepsilon^*) \approx -\log_2 \mu_{\theta^*}(x) + \frac{1}{2}\log_2 J(\theta^*) + \text{cst}. \tag{3.16}$$

Essentially, the idea is that if $\theta - \theta^* < \varepsilon^*$, and if we denote by x the data, the difference between $P_\theta(x)$ and $P_{\theta^*}(x)$ is not significant enough to justify using more bits to improve the precision of θ. Another possible way to look at this is that we cannot distinguins θ from θ^*, in the sense that it is hard to tell if the data have been sampled from one distribution or the other.

3.1.2 The i.i.d. case: confidence intervals and Fisher information

Let us now consider the i.i.d. case: all the x_i are sampled from the same probability distribution, and we have $-\log_2 \mu_\theta(x) = \sum_i -\log_2 \alpha_\theta(x_i)$. In that case, $J(\theta) = \sum_{i=1}^{n} \frac{\partial^2}{\partial\theta^2} \ln \alpha_\theta(x_i)$, so it is roughly proportional to n, and ε^* is therefore proportional to $\frac{1}{\sqrt{n}}$: we find the classical confidence interval.

Moreover, if the x_i really follow α_θ, then

$$\mathbb{E}_{x\sim\alpha}(J(\theta)) = n\mathbb{E}_{x\sim\alpha}\left(\frac{\partial^2}{\partial\theta^2}(-\ln \alpha_\theta(x))\right) =: nI(\theta), \tag{3.17}$$

where $I(\theta)$ is the so-called Fisher information [Fis22]. We will often use the following approximation

$$J(\theta) \approx nI(\theta). \tag{3.18}$$

With this, we can rewrite equation (3.15) for the i.i.d. case, and we find that the optimal ε is approximately equal to:

$$\varepsilon^* \approx \frac{1}{\sqrt{n}} \frac{1}{\sqrt{I(\theta^*)}}. \tag{3.19}$$

By simply studying the optimal precision to use for a parameter from a coding perspective, we managed to recover confidence intervals and the Fisher information.

3.1.3 Link with model selection

Now that we have the optimal precision and the corresponding codelength, we can also solve certain model selections problems.

Consider for example you are playing heads or tails n times, but at the middle of the game, the coin (i.e. the parameter of Bernoulli's law) is changed. You are given the choice to encode a single θ, or θ_1 which will be used for the first half of the data, and θ_2 which will be used for the second half.

Let us compute the codelengths corresponding to these two models. If we denote by x all the data, by x_1 the first half of the data, and by x_2 the second half of the data, we find, by combining equations (3.16) and (3.18):

$$l_1 = -\log_2 \mu_\theta(x) + \frac{1}{2}\log_2 I(\theta) + \frac{1}{2}\log_2(n) + \text{cst}, \tag{3.20}$$

$$l_2 = -\log_2 \mu_{\theta_1}(x_1) - \log_2 \mu_{\theta_2}(x_2) + \frac{1}{2}\log_2 I(\theta_1) + \frac{1}{2}\log_2 I(\theta_2) + \frac{1}{2}\log_2(n/2) + \frac{1}{2}\log_2(n/2) + \text{cst} \tag{3.21}$$

$$= -\log_2 \mu_{\theta_1,\theta_2}(x) + \frac{2}{2}\log_2(n) + O(1). \tag{3.22}$$

It is easy to see that for a model with k parameters, we would have:

$$l_k = -\log_2 \mu_{\theta_1,\dots,\theta_k}(x) + \frac{k}{2}\log_2(n) + O(1). \tag{3.23}$$

Asymptotically, we obtain the Bayesian information criterion, which is often used. It could be interesting to use the non-asymptotic equation (3.21) instead, but the Fisher information is usually hard to compute.

It is also interesting to notice that using two-part codes automatically makes the corresponding coding suboptimal, since it reserves several codewords for the same symbol: coding θ_1 followed by x coded with P_{θ_1} yields a different code than θ_2 followed by x coded with P_{θ_2}, but these two codes

are codes for x. A solution to this problem is to set $P_\theta(x) = 0$ if there exists θ' such that $P_{\theta'}(x) > P_\theta(x)$, and renormalize. Then, for a given x, only the best estimator can be used to encode x. This yields the normalized maximum likelihood distribution (if we denote by $\hat{\theta}(x)$ the maximum likelihood estimator for x, $\mathrm{NML}(x) = \dfrac{P_{\hat{\theta}(x)}(x)}{\sum_x P_{\hat{\theta}(x)}(x)}$).

Another way to look at this problem is the following: consider as an example the simple case $\Theta = \{1, 2\}$. The encoding of θ corresponds to a prior q on Θ (for example, using one bit to distinguish P_1 from P_2 corresponds to the uniform prior $q(1) = q(2) = 0.5$).

The two-part code corresponds to using $\max(q(1)P_1, q(2)P_2)$ as our "probability distribution" to encode the data, but its integral is not equal to 1: we lose $\int_X \min(q(1)P_1(x), q(2)P_2(x))dx$, which makes the codelengths longer. Consequently, it is more interesting to directly use the mixture $q(1)P_1 + q(2)P_2$ to encode the data when it is possible, because *all* codewords will then be shorter. We are thus naturally led to Bayesian models.

3.2 Bayesian models, Jeffreys' prior

3.2.1 Motivation

We need a reasonable default prior q for the probability distribution (3.10):

$$P_{\mathcal{F}}^{\mathrm{Bayes}}(x) = \int_\Theta q(\theta)\mu_\theta(x)\mathrm{d}\theta. \tag{3.24}$$

A naive choice for a default prior is "the uniform prior" (i.e. $q(\theta) = \mathrm{cst}$).

Sadly, *the* uniform prior on a family of probability distributions can be ill-defined.

Consider for example $\mathcal{B}$ the family of Bernoulli distributions.

If P_θ is the Bernoulli distribtion of parameter θ, and Q_θ is the Bernoulli distribtion of parameter θ^{100}, then $\{P_\theta, \theta \in [0, 1]\} = \{Q_\theta, \theta \in [0, 1]\} = \mathcal{B}$, but most of the time, the uniform prior on the family (Q_θ) will select a Bernoulli distribution with a parameter close to 0 (θ is picked uniformly in $[0, 1]$, so $\theta^{100} < 0.1$ with probability $0.1^{\frac{1}{100}} \gtrsim .97$).

This shows that a uniform prior depends not only on our family of probability distributions, *but also on its parametrization*, which is an arbitrary choice of the user.

Any reasonable default prior should be invariant by reparametrization of the family $\{P_\theta, \theta \in \Theta\}$ (else, there would be as many possible priors as there are parametrizations), and Jeffreys' prior [Jef61], constructed below, does have this property.

3.2.2 Construction

Consider we are sending a message of length n with a two-part code.

Equation (3.19) (about the optimal coding precision) shows that when coding a given message with a two-part code, only a finite number of elements of Θ will actually be used in the first part of the coding, each θ_k being used for all $\theta \in I_k$; and it also shows that the length of I_k is $\frac{1}{\sqrt{n}} \frac{1}{\sqrt{I(\theta_k)}}$.

The fact that all elements of I_k are encoded the same way means that we will not distinguish different θ in the same I_k, and in practice, we will only use the m different θ_k corresponding to each interval.

Consequently, a reasonable procedure to pick a θ would be to start by picking some $k \in [1, m]$, and then, pick $\theta \in I_k$ uniformly.

It is easy to see that the probability distribution q_n corresponding to this procedure is given by the density:

$$q_n(\theta) = K_n \sqrt{I(\theta_{k(\theta)})}, \tag{3.25}$$

where K_n is a normalization constant, and $k(\theta)$ is defined by the relation $\theta \in I_{k(\theta)}$.

Now, if $n \to \infty$, it can be proved[2] that (q_n) converges to the probability distribution q given by the density:

$$q(\theta) = K \sqrt{I(\theta)}, \tag{3.26}$$

where K is a normalization constant.[3] Also notice that sometimes (for example with the family of Gaussian distributions on $\mathbb{R}$), Jeffreys' prior cannot be normalized, and consequently, cannot be used.

We now have a reasonable default prior, and we are going to use it to predict the next element of a sequence of zeros and ones.

3.2.3 Example: the Krichevsky–Trofimov estimator

We introduce the following notation:

Notation 3.4. *We denote by P_θ the Bernoulli distribution of parameter θ, and we define $\mathcal{B} := \{P_\theta, \theta \in [0, 1]\}$.*

Suppose we are trying to learn a frenquency on the alphabet $X = \{c, d\}$. For example, given the message *ccc*, we want $P(x_4 = c)$.

A first approach is using the maximum likelihood (ML) estimator:

$$P^{\mathrm{ML}}(x_{n+1} = c | x_1, ..., x_n) = \frac{\sum_{i=1}^{n} \mathbb{1}_{x_i = c}}{n} = \frac{\text{number of c in the past}}{\text{number of observations}}. \tag{3.27}$$

[2] The important idea is that $\theta_{k(\theta)} \to \theta$ when $n \to \infty$, since the length of all the intervals I_k tends to 0.

[3] In dimension larger than 1, (3.26) becomes $q(\theta) = K \sqrt{\det I(\theta)}$.

This has two drawbacks: ML assigns the probability 0 to any letter that has not been seen yet (which is a major problem when designing codes, since probability 0 corresponds to infinite codelength), and is undefined for the first letter.

Let us now consider the Bayesian approach with Jeffreys' prior.

Lemma 3.5. *The Fisher metric on $\mathcal{B}$ is given by*

$$I(\theta) = \frac{1}{\theta(1-\theta)}, \tag{3.28}$$

and Jeffreys' prior is given by

$$q(\theta) = \frac{1}{\pi}\frac{1}{\sqrt{\theta(1-\theta)}} \tag{3.29}$$

Proof. It is a straightforward calculation. We have $P_\theta(0) = 1-\theta$, $P_\theta(1) = \theta$, so $\frac{\partial^2 \ln P_\theta(0)}{\partial\theta^2} = \frac{-1}{(1-\theta)^2}$ and $\frac{\partial^2 \ln P_\theta(1)}{\partial\theta^2} = \frac{-1}{\theta^2}$. Finally:

$$\begin{aligned} I(\theta) &= -\frac{\partial^2 \ln P_\theta(0)}{\partial\theta^2}P_\theta(0) - \frac{\partial^2 \ln P_\theta(1)}{\partial\theta^2}P_\theta(1) && (3.30)\\ &= \frac{1}{(1-\theta)^2}(1-\theta) + \frac{1}{\theta^2}\theta && (3.31)\\ &= \frac{1}{1-\theta} + \frac{1}{\theta} && (3.32)\\ &= \frac{1}{\theta(1-\theta)}, && (3.33) \end{aligned}$$

and we have the Fisher information. The only difficulty to compute Jeffreys' prior is the normalization constant, which is

$$K := \int_0^1 \frac{\mathrm{d}\theta}{\sqrt{\theta(1-\theta)}}, \tag{3.34}$$

and the substition $\theta = \sin^2 u$ yields

$$K = \int_0^{\pi/2} \frac{2\sin u \cos u \mathrm{d}u}{\sqrt{\sin^2 u \cos^2 u}} = \pi. \tag{3.35}$$

□

We have therefore, by using Jeffreys' prior in (3.10):

$$P^{\text{Jeffreys}}(x_{n+1}|x_1,...,x_n) = \frac{1}{\pi}\int_0^1 \frac{1}{\sqrt{\theta(1-\theta)}} P_\theta(x_1,...,x_n)P_\theta(x_{n+1}|x_1,...,x_n)\mathrm{d}\theta. \tag{3.36}$$

This is actually easy to compute, thanks to the following proposition (proved for example in [Fin97]):

Proposition 3.6. *Let $\alpha, \beta > 0$. If $q(\theta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$, then*

$$P^{\text{Bayes}}(x_{n+1} = c|x_1, ..., x_n) = \frac{\alpha + \sum_{i=1}^{n} \mathbb{1}_{x_i=c}}{\alpha + \beta + n}, \tag{3.37}$$

where

$$P^{\text{Bayes}}(x) = \int_{\Theta} q(\theta)\mathcal{B}_\theta(x)\mathrm{d}\theta \tag{3.38}$$

is the Bayesian prediction with Bernoulli distributions and the prior q on the parameter θ of the distributions.

In other words, using such a prior is equivalent to using ML, with fictional observations (c occuring α times and d occuring β times).[4]

In particular, for Jeffreys' prior, we have $\alpha = \beta = \frac{1}{2}$, so

$$P^{\text{Jeffreys}}(x_{n+1} = c|x_1, ..., x_n) = \frac{\frac{1}{2} + \sum_{i=1}^{n} \mathbb{1}_{x_i=c}}{n+1} = \frac{\frac{1}{2} + \text{number of c in the past}}{1 + \text{number of observations}}. \tag{3.39}$$

Consequently, we introduce the following notation: $\text{KT}(x, y) := \dfrac{\frac{1}{2} + x}{1 + x + y}$.

This estimator is called "Krichevsky–Trofimov estimator" (KT, [KT81]), and it can be proved ([WST95],[VNHB11]) that, when using it instead of knowing the "real" θ, no more than $1 + \frac{1}{2}\log(n)$ bits are wasted.

We can also define the generative model corresponding to Jeffreys' prior. It is easy to prove (by induction on the total number of symbols) that the probability assigned to any given sequence $(x_1, ..., x_{a+b})$ with a zeros and b ones is equal to

$$P_J(a, b) := \frac{\left(\prod_{0\leqslant i<a}(i + \frac{1}{2})\right)\left(\prod_{0\leqslant i<b}(i + \frac{1}{2})\right)}{\prod_{0\leqslant i<a+b}(i+1)} = \prod_{0\leqslant i<a} \text{KT}(i, 0) \prod_{0\leqslant j<b} \text{KT}(a, j), \tag{3.40}$$

with the convention that the empty product is equal to 1 (i.e. $P_J(0, 0) = 1$).

In particular, we can see that the probability of a sequence $(x_1, ..., x_n)$ depends only on the *number* of c and d in $(x_1, ..., x_n)$, and not on the order, and P_J satisfies the relations, $P_J(a + 1, b) = P_J(a, b)\text{KT}(a, b)$ and $P_J(a, b + 1) = P_J(a, b)(1 - \text{KT}(a, b))$.

It is also useful to remark that $P_J(0, 0) = 1$ and $P_J(1, 0) = P_J(0, 1) = \frac{1}{2}$.

In practice, Jeffreys' prior is not efficient when the optimal θ is on the boundary: for instance, when encoding text, most punctuation symbols

[4] Distributions satisfying $q(\theta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$ are called *beta* distributions. The simplification in this proposition is due to the fact that if we have a beta prior for a Bernoulli distribution, then the posterior will also be a beta distribution. The beta distributions are called *conjugate priors* to Bernoulli distributions.

are always followed by a space. The corresponding new prior for Bernoulli distributions is given by

$$\frac{1}{2}q + \frac{1}{4}\delta_0 + \frac{1}{4}\delta_1, \tag{3.41}$$

where q is Jeffreys' prior on $\mathcal{B}$, and δ_i is the Dirac distribution at i. This new prior is called a "zero-redundancy estimator" [RK05].

3.3 Context tree weighting

In this section, we quickly present context tree weighting, a Bayesian text prediction model combining *all* visible Markov models of any finite order, see [WST95] and [Wil94] for more information. In part II, we describe this algorithm and prove its fundamental property[5] in a more elegant way (Section 4.3.4.3 in particular), and we extend it to other applications.

We start by showing a simple way to describe *one* visible Markov model.

3.3.1 Markov models and full binary trees

Learning a frequency on the alphabet $X = \{c, d\}$ corresponds to restricting ourselves to the family of Bernoulli distributions, which can be thought of as the family of Markov models of order 0.

Still working on the alphabet $\{c, d\}$, we are going to present the context tree weighting algorithm, which uses the family of all visible Markov models.

Notice in particular that we can use Markov models of *variable* order, for example: "if the last symbol was c, predict c with probability 0.7, if the last symbol was d, then look at the next-to-last symbol: if it was a c then predict c with probability 0.5, else predict c with probability 0.3."

A Markov model can be described by a full[6] binary tree the following way:

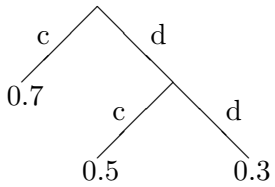


Figure 3.1: Tree corresponding to the example above

For a given tree, the contexts we are interested in are the leaves of the tree: with the example above, the leaf labeled 0.7 corresponds to the context "c?", the leaf 0.5 corresponds to the context "cd?" and the leaf 0.3

[5] It does compute the advertised Bayesian mixture.

[6] All nodes have either 0 or 2 children.

corresponds to "*dd*?" (notice in particular that contexts are read *backwards* in the tree).

Suppose for now that we have such a tree. It is easy to see that there is exactly one context appearing in the tree and corresponding to a suffix of the past observations: we output c with probability equal to the value of the leaf corresponding to this context.

Our model can therefore be decomposed as a finite full binary tree T (the structure of the tree), and the ordered set $\theta_T := (\theta_s)$ of the labels on the leaves, and we will try to estimate it online.

Now that we are able to describe efficiently a Markov model, we can go back to our main problem: prediction using universal probability distributions on all visible Markov models.

3.3.2 Prediction for the family of visible Markov models

The distribution we would like to use for prediction is (3.4), with $\Theta = \{(T, \theta_T), T \text{ finite binary tree}, \theta_T \in [0,1]^{\text{number of leaves of } T}\}$:

$$P_{\mathcal{F}} = \sum_{\theta \in \Theta} 2^{-K(\theta)} \mu_\theta,$$

where μ_θ is given by the previous subsection. $K(\theta)$ remains problematic, but we can decompose $\sum_\theta 2^{-K(\theta)}$ into $\sum_T 2^{-K(T)} \sum_{\theta_T} 2^{-K(\theta_T)}$, and then:

- A full binary tree T can be described by as many bits as it has nodes, by labelling internal nodes by 1 and leaves by 0, and read breadth-first (our example would give the code 10100).
- We can use Jeffreys' prior for θ_T.

With these new approximations, we have:

$$P_{\mathcal{F}} = \sum_{T \text{ full binary trees}} 2^{-\#\text{nodes}(T)} \int_{[0,1]^{\#\text{leaves}(T)}} \frac{\mathrm{d}\theta_{s_1}}{\pi\sqrt{\theta_{s_1}(1-\theta_{s_1})}} \cdots \frac{\mathrm{d}\theta_{s_l}}{\pi\sqrt{\theta_{s_l}(1-\theta_{s_l})}} P_{(T,(\theta_i))}, \tag{3.42}$$

where θ_i denotes the parameter of the i-th leaf. (3.42) can be interpreted as a double mixture over the full binary trees and over the parameter values.

Now, we have to compute (3.42). This sum has an infinite number of terms (and even if we restrict ourselves to a finite depth D, the number is still exponential in D). However, it is possible to compute it efficiently and *exactly*.

3.3.3 Computing the prediction

The general idea is to maintain the tree of all observed contexts, and each node s will weight the choices "using s as a context", and "splitting s into

the subonctexts cs and ds" by using information from deeper nodes (more precisely, the number of times a c or a d has been written in a given context).

3.3.3.1 Bounded depth

Here, we restrict ourselves to Markov models of depths at most D. For simplicity, we will not try to predict the D first symbols.

Notice that in that case, the cost to describe a tree can be modified: since a node at depth D is automatically a leaf, it is no longer necessary to use bits to describe them. If we denote by T_D the complete binary tree of depth D, and by $n_D(T)$ the number of nodes at depths less than D of a tree T, then the probability distribution corresponding to this model is

$$P_{\mathcal{F}} = \sum_{T \text{ subtree of } T_D} 2^{-n_D(T)} \int_{[0,1]^{\#\text{leaves}(T)}} \frac{\mathrm{d}\theta_{s_1}}{\pi\sqrt{\theta_{s_1}(1-\theta_{s_1})}} \dots \frac{\mathrm{d}\theta_{s_l}}{\pi\sqrt{\theta_{s_l}(1-\theta_{s_l})}} P_{(T,(\theta_i))}, \tag{3.43}$$

Notation 3.7. *We will denote each node by the suffix corresponding to it (for example, the root is ε, the node labeled 0.7 in Figure 3.1 is c, etc...), and at each node s, we will maintain two numbers c_s and d_s, respectively the number of times a c and a d have been observed after the context s (starting the count at x_1). Notice that we have the relation $c_s = c_{1s} + c_{0s}$.*[7]

Now, let us look at equation (3.43) more closely. The goal of the following lemma and its corollary is to write equation (3.43) in a way that will allow to compute it efficiently.

Firstly, we can see that

Lemma 3.8. *If $c_s(k)$ and $d_s(k)$ are respectively the number of times a c and a d have been observed after the context s from x_1 to x_k, we have:*

$$P_{(T,(\theta_{s_i}))}(x_1, ..., x_n) = \prod_{s \text{ leaves of T}} \theta_s^{c_s(n)} (1-\theta_s)^{d_s(n)}, \tag{3.44}$$

Proof. By induction. $P_{(T,(\theta_{s_i}))}(x_1) = \theta_{s_T(Lx_1)}^{\mathbb{1}_{x_1=c}} (1-\theta_{s_T(Lx_1)})^{\mathbb{1}_{x_1=d}}$, where $s_T(x)$ is the only leaf of T corresponding to a suffix of x, which is what we want.

Now,

$$\begin{aligned} P_{(T,(\theta_{s_i}))}(x_1, ..., x_{k+1}) =& P_{(T,(\theta_{s_i}))}(x_{k+1}|x_1, ..., x_k) P_{(T,(\theta_{s_i}))}(x_1, ..., x_k) \\ =& \theta_{s_T(Lx_1...x_k)}^{\mathbb{1}_{x_{k+1}=c}} (1-\theta_{s_T(Lx_1...x_k)})^{\mathbb{1}_{x_{k+1}=d}} \prod_{s \text{ leaves of T}} \theta_s^{c_s(k)} (1-\theta_s)^{d_s(k)}, \end{aligned}$$

[7] Unless s is a prefix of the word we are reading. In practice, as we will see in the next section, this problem can be solved by adding a new character § indicating the beginning of the message: $c_s = c_{0s} + c_{1s} + c_{\S s}$ is *always* true.

and since for all s for $\lambda \in \{c, d\}$, $\lambda_s(k+1) = \lambda_s(k) + \mathbb{1}_{x_{k+1}=\lambda}\mathbb{1}_{s=s_T(Lx_1...x_k)}$, we find

$$P_{(T,(\theta_{s_i}))}(x_1, ..., x_{k+1}) = \prod_{s \text{ leaves of T}} \theta_s^{c_s(k+1)}(1-\theta_s)^{d_s(k+1)}, \tag{3.45}$$

which is what we wanted. □

Consequently, the integral in (3.43) can be computed with Proposition 3.6:

Corollary 3.9. *We have:*

$$\int_{[0,1]^{\#\text{leaves}(T)}} \frac{\mathrm{d}\theta_{s_1}}{\pi\sqrt{\theta_{s_1}(1-\theta_{s_1})}} ... \frac{\mathrm{d}\theta_{s_l}}{\pi\sqrt{\theta_{s_l}(1-\theta_{s_l})}} P_{(T,(\theta_i))}(x_1, ..., x_n) = \prod_{u \text{ leaves of } T} P_J(c_u, d_u) \tag{3.46}$$

Proof. It is a consequence of Lemma 3.8 and of the fact that $P_J(a, b)$ is the probability of seeing a sequence containing respectively a and b times the letters c and d with Jeffreys' prior. □

Equation 3.43 can therefore be rewritten:

$$P_{\mathcal{F}}(x_1, ..., x_n) = \sum_{T \text{ subtree of } T_D} 2^{-n_D(T)} \prod_{u \text{ leaves of } T} P_J(c_u, d_u). \tag{3.47}$$

Under this form, $P_{\mathcal{F}}$ can be computed recursively, starting by its leaves, by the following lemma:

Lemma 3.10. *If we assign to each node s the following probability:*

$$P^s := \begin{cases} P_J(c_s, d_s) & \text{if } |s| = D \\ \frac{1}{2}P_J(c_s, d_s) + \frac{1}{2}P^{cs}P^{ds} & \text{if } |s| < D \end{cases} \tag{3.48}$$

then, for any node s, P^s satsifies:

$$P^s = \sum_{T \text{ subtree of } T_D \text{ with root } s} 2^{-n_D(T)} \prod_{u \text{ leaves of } T} P_J(c_u, d_u). \tag{3.49}$$

Notice that by definition, s is automatically a suffix of all the u in the equation above.

Proof. By induction. If $|s| = D$, it is clearly true. Suppose now that (3.49) holds for all nodes deeper than s. We have, by definition:

$$\begin{aligned}
P^s =& \frac{1}{2} P_J(c_s, d_s) + \frac{1}{2} P^{cs} P^{ds} \\
=& \frac{1}{2} P_J(c_s, d_s) + \frac{1}{2} \left(\sum_{T \text{ subtree of } T_D \text{ with root } cs} 2^{-n_D(T)} \prod_{u \text{ leaves of } T} P_J(c_u, d_u) \right) \\
& * \left(\sum_{T \text{ subtree of } T_D \text{ with root } ds} 2^{-n_D(T)} \prod_{u \text{ leaves of } T} P_J(c_u, d_u) \right)
\end{aligned}$$

The first term corresponds to the subtree composed only of the root, and the second term corresponds to all other subtrees, described by the part on the left of the root and the part of the right of the root. In other words:

$$P^s = 2^{-1} P_J(c_s, d_s) + \sum_{\substack{T \text{ subtree of } T_D \text{ with root } cs \\ T' \text{ subtree of } T_D \text{ with root } ds}} 2^{-1-n_D(T)-n_D(T')} \prod_{u \text{ leaves of } T \text{ or } T'} P_J(c_u, d_u), \tag{3.50}$$

which is what we want.

□

As an immediate corollary, we have:

$$P^\varepsilon = \sum_{T \text{ subtree of } T_D \text{ with root } \varepsilon} 2^{-n_D(T)} \prod_{u \text{ leaves of } T} P_J(c_u, d_u), \tag{3.51}$$

which is exactly equation (3.47).

Let us now consider the general case.

3.3.3.2 Generalization

The method presented above has two shortcomings: the fact that we cannot predict the first D symbols, and the bounded depth.

The first one can be solved by adding a "beginning of the message" character §. The corresponding trees are then ternary, and since undefined symbols can only occur at the beginning of a word, any suffix starting with § is a leaf.

Now, to work with unbounded depth, we simply maintain the tree T_n of all suffixes seen up to time n (i.e: all factors of the word $\S, x_1, ..., x_{n-1}, x_n$. T_n is therefore exactly of depth $n+1$), and, at each node s of T_n, the counts c_s, d_s, and $\S_s$.

To compute the corresponding distribution, we replace the P^s in (3.48) by:

$$P^s := \begin{cases} \frac{1}{2} & \text{if} \quad s \text{ is a leaf of } T_n \\ \frac{1}{2}P_J(c_s, d_s) + \frac{1}{2}P^{cs}P^{ds}P^{\S s} & \text{else,} \end{cases} \tag{3.52}$$

with the convention that $P^{\lambda s} = 1$ if λs has never appeared yet (for $\lambda = c, d, \S$), and we use P^ε again as our probability distribution.

It can be proved that this algorithm achieves entropy (i.e. optimal coding length) for arbitrary-depth tree sources (Theorem 3 in [Wil94]).

We can now describe the actual algorithm.

3.3.4 Algorithm

Suppose that we have $P_{\mathcal{F},n}(x_n|x_1, ..., x_{n-1})$, with the corresponding tree T_{n-1} (the tree containing all suffixes in $x_1, ..., x_{n-1}$).

When we read x_n, we update T_{n-1} by adding to it the suffix corresponding to the branch $(x_1, ..., x_n)$. We add at most n nodes. Now, we compute the new P^s. Since the only modified P^s are those corresponding to a suffix of $(x_1, ..., x_n)$, the complexity is $O(n)$.

We can therefore read $P(x_1, ..., x_n)$ at the root. Now, add the node $x_1, ..., x_n, c$ to compute $P(x_1, ..., x_n, c)$ at the root (the computation is again $O(n)$), and the prediction is $P_{\mathcal{F},n+1}(x_{n+1}|x_1, ..., x_n) = \frac{P_{\mathcal{F},n+1}(x_1,...,x_{n+1})}{P_{\mathcal{F},n}(x_1,...,x_n)}$.

The complexity of this algorithm is $O(n^2)$, whereas the bounded version has complexity $O(nD)$, where, n is the length of the data we are compressing and D is the depth bound. However, if the data is "sufficiently random", the complexity of the unbounded version is only $O(n \log_2 n)$ when coded properly[8], which makes it usable in general.

[8]The algorithm described here is always $O(n^2)$, but it can be improved by using the fact that some points are "equivalent" (for example, we can stop going deeper in the tree once we find a suffix that has appeared only once).

Part II

Expert Trees

The Context Tree Weighting algorithm, detailed in Section 4.3.1, is a well-known online text compression algorithm which computes a very large Bayesian mixture of models with a suitable prior (all finite visible Markovian models) efficiently by starting with complex models (i.e. models using a long context for the current data point), and recursively comparing them to (just) simpler models, up to the root of the tree, which will give the final prediction. In other words, each node of the context tree computes the (Bayesian) weight of the options "being open" (i.e. using more sophisticated experts) or "being closed".

This general idea can actually be applied to a much larger class of problems including regression, and density estimation: in Chapter 4, we give an adequate formal context to define a generic Context Tree Weighting algorithm: we define *experts*, which are entities that make predictions, and we give different ways of combining them.

Moreover, it has been noted in [vEGdR12] that Bayesian mixtures have a lot of "intertia": complex experts need a long time to estimate their parameters correctly, so by design, they make poor predictions in the beginning, which hurts their Bayesian posterior. The Bayesian mixture starts using them again only once their new predictions offset their slow start.

This issue is adressed (still in [vEGdR12]) by introducing *Switch distributions*:

- From a predictive point of view, at each step, the switch distribution takes a small part of the posterior and redistributes it, thus allowing complex models to quickly become relevant once they start making good predictions.
- From a generative point of view, while a regular Bayesian mixture corresponds to an average on the different models in it, the switch distribution is an average on *sequences* of models. In particular, we can expect that sequences with models of increasing complexity to perform very well.

Since the Context Tree Weighting algorithms contain infinitely many models of arbitrary complexity, it seems reasonable to try to use Switch distribution in context trees. It is not possible to compute the switch distribution corresponding to all finite visible Markovian models, but it is possible to use switch distributions locally at each node, for the open/closed choice.

This approach has been tried for text compression in [VNHB11], with promising results. However, the switch distributions considered in [VNHB11] switch infinitely many times with probability one, while we expect that the most interesting sequences on a given node are of the form "closed until time n (so that the more specialized experts get enough data), open afterwards" (with possibly $n = \infty$). Consequently, we try using the distributions in

[vEGdR12], which give a strictly positive wieght to any of these sequences instead.

Chapter 5 is a proof of concept for using these switch distributions in context trees: we show that in the case where local experts are fitting a Gaussian with variable mean and variance, the Context Tree Swiching algorithm outperforms the Context Tree Weighting algorithm on any Lipschitz function, in the sense that it assigns a higher probability to the actually observed data (i.e. the comparison criterion is the log-loss).

Finally, these results are illustrated by numerical experiments in Chapter 6.

Chapter 4

Expert trees: a formal context

We begin by giving a formal context which will enable us to describe the generalized context tree weighting and context tree switching algorithms easily.

Experts are a generalization of Prequential Forecasting Systems introduced in [Daw84], that allow for an elegant expression of the Context Tree Weighting algorithm.

The general idea is that we have the choice between a "simple" expert, and two (or more) more complex experts, each of these experts making recursively the same choice. For example, in the text compression setting, as seen in Section 3.3, an expert chooses between directly making its prediction and asking an expert with longer context, while in the regression setting, one could choose between a local expert on $[0, 1)$ and two experts, on $[0, 0.5)$ and $[0.5, 1)$.

We will consider explicitly three different problems to illustrate the definitions:

- Text compression: We are trying to predict the character $x_n \in X$ knowing the previous characters $x^{n-1} \in X^*$.
- Density estimation: We are trying to predict where the point $x_n \in X$ will appear, knowing the previous points x^{n-1}.
- Regression: Given some unknown, possibly noisy function $f : X \to Y$, we are trying to predict $f(x_n)$, knowing the $(x_k, f(x_k))$ for $k < n$.[1]

In Section 4.1, we give a formal definition of experts, and in Section 4.2, we define basic operations to build new experts from existing ones. In particular, the Switching combination of experts is defined in Section

[1] As seen in part I, we do note care about the "existence" of such a function, we are simply trying to compress the data using different models.

4.2.3. Finally, we define Expert Trees in Section 4.3, which allows for the description of a generic context tree weighting/switching algorithm.

4.1 Experts

Keeping in mind that a prediction is a probability distribution (as seen in part I), we define experts as applications matching previous data to a prediction:

Definition 4.1. *An expert from X to Y is an application $\mathcal{E}$ from a subset D of $(X \times Y)^* \times X$ to probability distributions on a non empty subset T of Y.*

D is called the domain *of $\mathcal{E}$, and is noted* $\mathrm{Dom}(\mathcal{E})$.

T is called the target *of $\mathcal{E}$, and is noted* $\mathrm{Tar}(\mathcal{E})$.

If X is a singleton[2]*, we will identify $X \times Y$ with Y, and we will simply write "an expert on Y".*

This is a generalization of Prequential Forecasting Systems [Daw84], in the sense that Prequential Forecasting Systems are experts with domain $(X \times Y)^* \times X$ and target Y. [3]

Intuitively:

- $\mathcal{E}(\mathbf{z})(z')$ is the probability assigned to z' after having seen $\mathbf{z}$.
- An expert issues a prediction only if the previous data falls in its domain [4], and the prediction is conditioned on the new data falling in its target ("if y is in my target, then I think it follows the probability distribution p").

For formal simplicity, we will only consider the cases Y discrete, or Y continuous but the probability distribution issued by the experts have a density (and in that case, $\mathcal{E}(\mathbf{x})(y)$ is the density at y). This leads us to the following notation:

Notation 4.2. *Let X, Y be two sets. If $\mathcal{E}$ is an expert from X to Y, $\mathbf{z}, \mathbf{z}' = (\mathbf{x}', \mathbf{y}') \in (X \times Y)^*$, we write:*

$$\mathcal{E}(\mathbf{z}'|\mathbf{z}) := \prod_{\substack{1 \leqslant i \leqslant |\mathbf{z}'| \\ (\mathbf{z} \cdot (\mathbf{z}')^{i-1}, x_i') \in \mathrm{Dom}(\mathcal{E}) \\ y_i' \in \mathrm{Tar}(\mathcal{E})}} \mathcal{E}(\mathbf{z} \cdot (\mathbf{z}'^{<i}), x_i')(y_i'). \tag{4.1}$$

[2] This will happen for text compression and denstiy estimation.

[3] *Specialists* ([FSSW97], [KAW12]) could also be seen as experts with target Y (the specialist being awake iff the previous data falls in its domain). However, the main point of defining specialists is the combined prediction of a bunch of specialists (a specialist that is asleep issues the same prediction as the average of the awake specialists), whereas the experts defined here are not penalized for predictions that are out of their domain.

[4] Notice that there is nothing in the definition preventing an expert from using data outside of its domain.

Moreover, if $x \in X$, $y \in Y$, $\mathbf{z}$,$\mathbf{z}' \in (X \times Y)^*$, we also write:

$$\mathcal{E}((y, \mathbf{z}')|(\mathbf{z}, x)) := \mathcal{E}((x, y, \mathbf{z}')|\mathbf{z}) \tag{4.2}$$

The following proposition justifies using $\mathcal{E}(\mathbf{z}'|\mathbf{z})$ for $\mathcal{E}(\mathbf{z})(\mathbf{z}')$:

Proposition 4.3. *If $\mathcal{E}$ is an expert from X to Y, $\mathbf{z}, \mathbf{z}', \mathbf{z}'' \in (X \times Y)^*$, we have:*

$$\mathcal{E}(\mathbf{z}' \cdot \mathbf{z}''|\mathbf{z}) = \mathcal{E}(\mathbf{z}'|\mathbf{z})\mathcal{E}(\mathbf{z}''|\mathbf{z} \cdot \mathbf{z}'). \tag{4.3}$$

In particular, if $\mathcal{E}(\mathbf{z}|\varnothing) \neq 0$,

$$\mathcal{E}(\mathbf{z}'|\mathbf{z}) = \frac{\mathcal{E}(\mathbf{z} \cdot \mathbf{z}'|\varnothing)}{\mathcal{E}(\mathbf{z}|\varnothing)}, \tag{4.4}$$

Proof. Immediate consequence of the definition of $\mathcal{E}(\mathbf{z}'|\mathbf{z})$. □

By definition, if $y \in Y \setminus \mathrm{Tar}(\mathcal{E})$, then $\mathcal{E}(y|\mathbf{x}) = 1$. In particular, $\mathcal{E}(.|\mathbf{x})$ is not a probability distribution on Y restricted to $\mathrm{Tar}(\mathcal{E})$. This is the reason the target of an expert *has to be defined explicitly.*

Moreover, by equation (4.4), an expert $\mathcal{E}$ that never issues a zero probability can also be defined by specifying $\mathcal{E}(\mathbf{z}|\varnothing)$ for $\mathbf{z} \in D \times T$ instead of $\mathcal{E}(y|\mathbf{z}, x)$ (or equivalently $\mathcal{E}((x, y)|\mathbf{z})$) for all $(\mathbf{z}, x) \in \mathrm{Dom}(\mathcal{E})$, $y \in \mathrm{Tar}(\mathcal{E})$.

Finally, in the particular case of an expert $\mathcal{E}$ on a set X, with $\mathrm{Tar}(\mathcal{E}) = X$ and $\mathrm{Dom}(\mathcal{E}) = X^*$, $\mathcal{E}(.|\varnothing)$ is a *probabilistic source* (as defined in [Gru07]) on X. Indeed, we can easily check that:

- For all $\mathbf{x} \in X^*$, $\sum_{y \in X} \mathcal{E}(\mathbf{x}y|\varnothing) = \sum_{y \in X} \mathcal{E}(\mathbf{x}|\varnothing)\mathcal{E}(y|\mathbf{x}) = \mathcal{E}(\mathbf{x}|\varnothing)$
- $\mathcal{E}(\varnothing|\varnothing) = 1$,

4.1.1 General properties

In this section, we define a "no peeking out" condition, ensuring that an expert only uses data in its domain and target, and a compatibility condition between experts. These two conditions will be necessary for the good implementation of the expert tree algorithms.

We say that an expert does not peek out if it only watches its domain: it updates its predictions only after making one.

Notation 4.4. *If $\mathbf{z} = (z_1, ..., z_n) \in A^*$ and $S \subset A^*$, we define:*

$$\mathbf{z} \cap A := (z_i | z^i \in S). \tag{4.5}$$

If $\mathcal{E}$ is an expert from X to y with domain D and target T and $\mathbf{z} = ((x_1, y_1), ..., (x_n, y_n)) \in (X \times Y)^n$, we define $\mathbf{z} \cap \mathcal{E}$ as the subword of $\mathbf{z}$ containing only the (x_i, y_i) for which $\mathcal{E}$ has been used. In other words:

$$\mathbf{z} \cap \mathcal{E} := \mathbf{z} \cap (D \times T) = \big((x_i, y_i) | (z^{<i}, x_i) \in D, y_i \in T\big) \tag{4.6}$$

Definition 4.5 (No peeking out)**.** *Let $\mathcal{E}$ be an expert from X to Y with domain D and target T. We say $\mathcal{E}$ does not peek out iff:*

$$\mathcal{E}(\mathbf{z}|\varnothing) = \mathcal{E}(\mathbf{z} \cap \mathcal{E}|\varnothing), \tag{4.7}$$

for all $\mathbf{z} \in (X \times Y)^*$

This condition will be used to ensure that the Context Tree Weighting predictions can be computed in a reasonable time.

The compatibility condition will be used to define infinite depth expert tree algorithms.

Definition 4.6 (Compatible experts)**.** *Let X, Y be two sets, and let $\mathcal{E}_1$, $\mathcal{E}_2$ be two experts from X to Y with same domain and same target. We say they are* compatible *if*

$$\mathcal{E}_1(\varnothing) = \mathcal{E}_2(\varnothing) \tag{4.8}$$

as applications $X \to Y$.

If $\mathbf{T}$ is an expert tree, we say that the experts in $\mathbf{T}$ are compatible if for any internal node s, we have: $\bigcup_{t \in c(s)} \mathcal{E}_t(\varnothing) = \mathcal{E}_s(\varnothing)$.

We now define several operations allowing us to create new experts from existing ones.

4.2 Operations with experts

In this section, we define the mixture of experts, the bayesian and switch combination of experts, the restriction of an expert, and the union of experts.

4.2.1 Fixed mixtures

The fixed mixture of n experts consists in averaging the prediction of these experts with fixed coefficients:

Definition 4.7. *Let $(\alpha_1, ..., \alpha_n) \in \mathbb{R}_+^n \setminus \{0, ..., 0\}$.*

The fixed mixture of n density experts on X with same domain D and target T with coefficients $\alpha_1, ..., \alpha_n$, with is the expert with domain D and target T defined by:

$$\text{mix}((\alpha_i \mathcal{E}_i)_{i=1,..,n})(z'|\mathbf{z}) := \frac{1}{\sum_i \alpha_i} \sum_i \alpha_i \mathcal{E}_i(z'|\mathbf{z}), \tag{4.9}$$

for $z' = (x', y')$, $(\mathbf{z}, x') \in D$, $y' \in T$.

Notice however that for $\mathbf{y} \in T^*$, $\text{mix}((\alpha_i \mathcal{E}_i)_{i=1,..,n})(\mathbf{y}|\mathbf{x})$ is not equal to $\frac{1}{\sum_i \alpha_i} \sum_i \alpha_i \mathcal{E}_i(\mathbf{y}|\mathbf{x})$ in general.

4.2.2 Bayesian combinations

Definition 4.8. *Let $\mathcal{E}_1, ..., \mathcal{E}_n$ be n experts from X to Y with the same domain D and target T such that for all $\mathbf{z} \in (X \times Y)^*$, there exists $i \in \{1, ..., n\}$ such that $\mathcal{E}_i(\mathbf{z}|\varnothing) \neq 0$.*

The Bayesian combination of $\mathcal{E}_1, ..., \mathcal{E}_n$ is the expert with domain D and target T defined by:

$$\text{Bayes}(\mathcal{E}_1, ..., \mathcal{E}_n)(z'|\mathbf{z}) := \frac{\sum_i \mathcal{E}_i(\mathbf{z}|\varnothing)\mathcal{E}_i(z'|\mathbf{z})}{\sum_i \mathcal{E}_i(\mathbf{z}|\varnothing)} = \frac{\sum_i \mathcal{E}_i(\mathbf{z} \cdot z'|\varnothing)}{\sum_i \mathcal{E}_i(\mathbf{z}|\varnothing)}, \tag{4.10}$$

for $z' = (x', y')$, $(\mathbf{z}, x') \in D$, $y' \in T$.

Under the same assumptions, we also define the Bayesian combination of $\mathcal{E}_1, ..., \mathcal{E}_n$ with prior $w_1, ..., w_n$ as:

$$\text{Bayes}\left((\mathcal{E}_1, w_1), ..., (\mathcal{E}_n, w_n)\right)(z'|\mathbf{z}) := \frac{\sum_i w_i \mathcal{E}_i(\mathbf{z}|\varnothing)\mathcal{E}_i(z'|\mathbf{z})}{\sum_i w_i \mathcal{E}_i(\mathbf{z}|\varnothing)}, \tag{4.11}$$

and the Bayesian combination of $(\mathcal{E}_\theta, w_\theta)_{\theta \in \Theta}$ as

$$\text{Bayes}\left((\mathcal{E}_\theta, w_\theta)_{\theta \in \Theta}\right)(z'|\mathbf{z}) := \frac{\int_\Theta w_\theta \mathcal{E}_\theta(\mathbf{z}|\varnothing)\mathcal{E}_\theta(z'|\mathbf{z})\mathrm{d}\theta}{\int_\theta w_\theta \mathcal{E}_\theta(\mathbf{z}|\varnothing)\mathrm{d}\theta}. \tag{4.12}$$

As an immediate consequence of the definition, we have:

Proposition 4.9. *For all $\lambda \in \mathbb{R}^*$, we have*

$$\text{Bayes}((\mathcal{E}_i, \lambda w_i)) = \text{Bayes}((\mathcal{E}_i, w_i)) \tag{4.13}$$

An important property of the Bayesian combination is that from a generative point of view, it is simply a sum:

Proposition 4.10. *We have, for all $\mathbf{z}, \mathbf{z}' \in (X \times Y)^*$:*

$$\text{Bayes}((\mathcal{E}_1, w_1), ..., (\mathcal{E}_n, w_n))(\mathbf{z}'|\mathbf{z}) = \frac{\sum_i w_i \mathcal{E}_i(\mathbf{z}|\varnothing)\mathcal{E}_i(\mathbf{z}'|\mathbf{z})}{\sum_i w_i \mathcal{E}_i(\mathbf{z}|\varnothing)}. \tag{4.14}$$

In particular, if the w_i sum to one:

$$\text{Bayes}((\mathcal{E}_i, w_i))(\mathbf{z}|\varnothing) = \sum w_i \mathcal{E}_i(\mathbf{z}|\varnothing) \tag{4.15}$$

Proof. By definition, we have:

$$\text{Bayes}((\mathcal{E}_1, w_1), ..., (\mathcal{E}_n, w_n))(\mathbf{z}'|\mathbf{z}) = \prod_{k=1}^{|\mathbf{z}'|} \text{Bayes}((\mathcal{E}_1, w_1), ..., (\mathcal{E}_n, w_n))(z'_k|\mathbf{z} \cdot \mathbf{z}'^{<k}) \tag{4.16}$$

$$= \prod_{k=1}^{|\mathbf{z}'|} \frac{\sum_i w_i \mathcal{E}_i(\mathbf{z} \cdot \mathbf{z}'^{\leqslant k}|\varnothing)}{\sum_i w_i \mathcal{E}_i(\mathbf{z} \cdot \mathbf{z}'^{<k}|\varnothing)} \tag{4.17}$$

$$= \frac{\sum_i w_i \mathcal{E}_i(\mathbf{z} \cdot \mathbf{z}'|\varnothing)}{\sum_i w_i \mathcal{E}_i(\mathbf{z}|\varnothing)}, \tag{4.18}$$

which is what we wanted. □

Corollary 4.11. *For any three experts $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{E}_3$ with same domain D and target T, and four weights w_1, w_2, w_3, $w_{2,3}$, we have:*

$$\text{Bayes}((\mathcal{E}_1, w_1), (\text{Bayes}((\mathcal{E}_2, w_2), (\mathcal{E}_3, w_3)), w_{2,3})) = \text{Bayes}((\mathcal{E}_1, w_1), (\mathcal{E}_2, w_{2,3}\frac{w_2}{w_2 + w_3}), (\mathcal{E}_3, w_{2,3}\frac{w_3}{w_2 + w_3})) \tag{4.19}$$

Proof. Immediate consequence of (4.15). □

Corollary 4.12. *Let $\mathcal{E}_1, ..., \mathcal{E}_n$ be n experts with the same domain D and target t such that for all $\mathbf{z} \in (X \times Y)^*$, there exists $i \in \{1, ..., n\}$ such that $\mathcal{E}_i(\mathbf{z}|\varnothing) \neq 0$.*

If the $\mathcal{E}_i$ do not peek out, then $\text{Bayes}_i(\mathcal{E}_i)$ do not peek out.

Proof. Immediate with (4.15). □

Following the specialist framework, a more general Bayesian combination could be defined by removing the condition that all the experts in the combination must have the same domain.

4.2.3 Switching

The Bayesian combination penalizes models with many parameters too harshly, because these parameters need to be estimated:

Consider that we are trying to predict a text actually generated with a (non degenerate) Markov model of order 2 using $\mathcal{B}$, a Bayesian combination of the experts $\mathcal{E}_1$ and $\mathcal{E}_2$, using visible Markov models of respective order 1 and 2. At the beginning, $\mathcal{E}_2$ will not have enough data to have a good estimate of its parameters, and will be inferior to $\mathcal{E}_1$. At some point t_0, $\mathcal{E}_2$ will start being better than $\mathcal{E}_1$, but its posterior weight at this time $w_2(t_0)$ will be much smaller than $w_1(t_0)$, so $\mathcal{B}$ will still behave like $\mathcal{E}_1$, until some time t_1, when w_2 catches up with w_1 (this is the "catch-up phenomenon" in [vEGdR12]).

Switching, which has been introduced in [vEGdR12], and also discussed in [KdR08] and [vE10], remedies this drawback: from a generative point of view, Bayes consists in choosing one model, and sticking with it, whereas Switch allows switching (hence the name) between models, and the posterior weights remain quick to compute.

4.2.3.1 Definition

We start with a concise way of writing sequences of experts:

Notation 4.13. *Given n experts $\mathcal{E}_1$,...,$\mathcal{E}_n$ with same domain D and target T, and $I = (i_1, ..., i_k, ...) \in [\![1, n]\!]^{\mathbb{N}^*}$, we denote by $\mathcal{E}_I$ the expert using $\mathcal{E}_{i_k}$ for its k-th prediction:*

$$\mathcal{E}_I(y|\mathbf{z}, x) = \mathcal{E}_{i_{\phi(\mathbf{z})}}(y|\mathbf{z}, x) \tag{4.20}$$

for all $y \in T$, $(\mathbf{z}, x) \in D$, where $\phi(\mathbf{z}) := 1 + |\mathbf{z} \cap (D \times T)|$

We define the switching combination of n experts as a Bayesian combination on *sequences* of experts:

Definition 4.14. *Let $\mathcal{E}_1$,...,$\mathcal{E}_n$ be experts with same domain D and target T, and let μ be a probability distribution on $\{1, ..., n\}^{\mathbb{N}}$. We define the expert* $\text{Switch}_\mu(\mathcal{E}_1, ..., \mathcal{E}_n)$ *by:*

$$\text{Switch}_\mu(\mathcal{E}_1, ..., \mathcal{E}_n)(\mathbf{z}|\varnothing) := \sum_{I \in \{1,...,n\}^{\mathbb{N}}} \mu(I)\mathcal{E}_I(\mathbf{z}|\varnothing) = \text{Bayes}_I(\mathcal{E}_I, \mu(I))(\mathbf{z}|\varnothing), \tag{4.21}$$

for all $\mathbf{z} \in D \times T$.

More generally, we have:

$$\text{Switch}_\mu(\mathcal{E}_1, ..., \mathcal{E}_n)(\mathbf{z}'|\mathbf{z}) = \sum_{I \in \{1,...,n\}^{\mathbb{N}}} \mu(I|\mathbf{z})\mathcal{E}_I(\mathbf{z}'|\mathbf{z}), \tag{4.22}$$

where $\mu(I|\mathbf{z}) := \dfrac{\mu(I)\mathcal{E}_I(\mathbf{z}|\varnothing)}{\sum_{J \in \{1,...,n\}^{\mathbb{N}}} \mu(J)\mathcal{E}_J(\mathbf{z}|\varnothing)}$ is the Bayesian posterior weight after $\mathbf{z}$ of I.

Proof. Direct application of Proposition 4.10. □

Notice that in particular, if $\mu(I) = 0$ for all non constant sequences I, then $\text{Switch}_\mu(\mathcal{E}_1, ..., \mathcal{E}_n)$ is simply a Bayesian combination of the $(\mathcal{E}_i)$. Of course, for most probability distributions μ, it is impossible to compute the posterior weights corresponding to (4.21).

However, under certain conditions on μ, the predictions of the switch experts can be computed in a reasonable time.

4.2.3.2 Computing some switch distributions: The forward algorithm

This section follows [KdR08] and [vEGdR12].

The most important condition for some switch distributions to be computed exactly is that the posterior weight of each expert at time $n+1$ must only depend of the weights at time n.

We start by describing a probability distribution on "switch parameters": sequences $(t^i, k^i)_{i_i n \{1,...,m\}}$ corresponding to sequences of experts starting with k_1, and switching at the t_i to the expert k_i ; we then pull this probability distribution back to actual sequences of experts:

Definition 4.15. *Let $\Xi = \{\mathcal{E}_1, ..., \mathcal{E}_n\}$ be a set of experts with domain D and target T.*

We call set of switch parameters on Ξ *the set:*

$$\Theta = \{(t^m, k^m) | 1 \leqslant m \leqslant \infty, k^m \in \Xi^m, t^m \in \mathbb{N}^m, 0 = t_1 < t_2 < ...\}.^{5} \quad (4.23)$$

We now define a surjective map from Θ to sequences of experts $\phi : \Theta \to \Xi^{\mathbb{N}}$ by: $\phi(\theta) := (\phi_1(\theta), \phi_2(\theta), ...)$, and:

$$\phi_n(t^m, k^m) = \mathcal{E}_{k_{\sup\{i \in \{1,...,m\} | t_i < n\}}}. \quad (4.24)$$

Any probability distribution π on Θ defines a probability distribution on $\Xi^{\mathbb{N}}$ by

$$\pi^*(A) = \pi(\phi^{-1}(A)), \quad (4.25)$$

for $A \subset \Xi^{\mathbb{N}}$.

In other words, ϕ sends the switch parameters (t^m, k^m) to the sequence using k_1 for $t \in]t_1, t_2]$, k_2 for $t \in]t_2, t_3]$ and so on. We now describe the actual prior that we will use, introduced in [vEGdR12].

Definition 4.16. The switch prior π_{Switch} *is the following distribution on switch parameters:*

$$\pi_{\text{Switch}}(t^m, k^m) = \pi_M(m)\pi_K(k_1) \prod_{i=2}^{m} \pi_\tau(t_i | t_i > t_{i-1})\pi_K(k_i), \quad (4.26)$$

where π_M is geometric with rate θ, π_K is an arbitrary probability distribution on Ξ, and π_τ is an arbitrary probability distribution on $\mathbb{N}^*$.

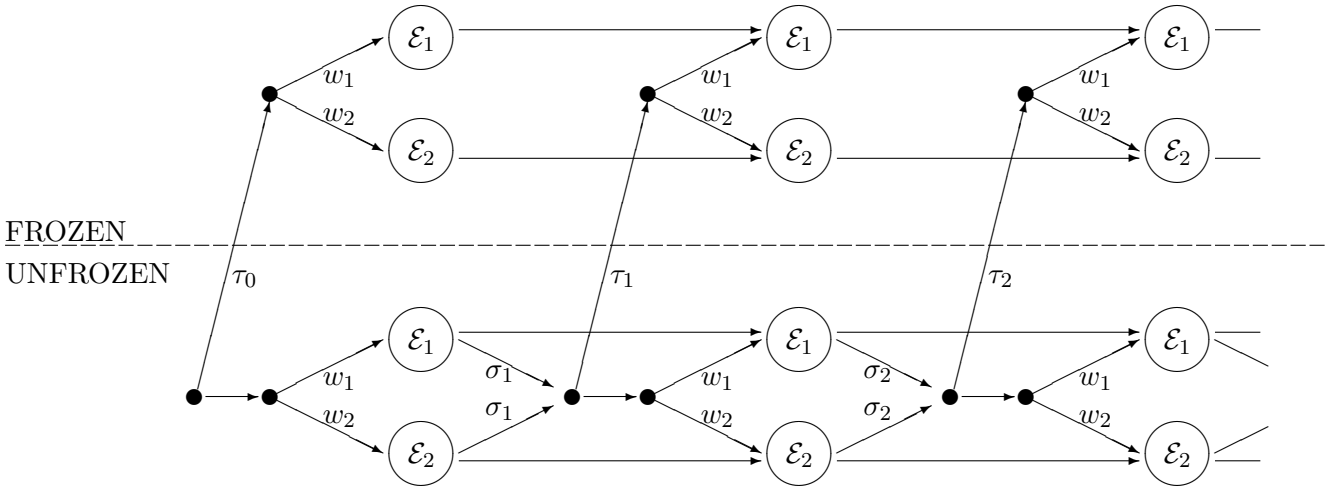
As described in [KdR08] and [vEGdR12], this probability distribution can be computed by using a hidden Markov model with the following parameters:

- $\Xi = \{\mathcal{E}_1, ..., \mathcal{E}_n\}$ a set of experts with domain D and target T.
- A probability distribution w_Ξ on Ξ (or on $\{1, ..., n\}$, by identifying $\mathcal{E}_i$ and i).
- Freezing probabilities $(\tau_n)_{n \in \mathbb{N}}$: a sequence of probability distributions on $\{0, 1\}$. By abuse of notation, we will write $\tau_n := \tau_n(1)$ the probability of freezing at time n.
- Switching probabilities $(\sigma_n)_{n \in \mathbb{N}^*}$: a sequence of probability distributions on $\{0, 1\}$. By abuse of notation, we will write $\sigma_n := \sigma_n(1)$ the probability of switching at time n.

and, the forward algorithm, described for example in [KdR08] computes the predictions for a HMM (Theorem 3 in [KdR08]).

[5] [KdR08] and [vEGdR12] only consider the case $m < \infty$.

Figure 4.1: The unfolded switch HMM, with two experts, with $w_i := w_\Xi(\mathcal{E}_i)$
(see also Figure 12 in [KdR08])



Algorithm 1 Forward algorithm from time n to time $n+1$, for the switch HMM

1. Observe z_n and compute posterior weights at time n.
for all $\mathcal{E} \in \Xi, * \in \{u, f\}$ **do**
$w(\mathcal{E}, *, n) \leftarrow w(\mathcal{E}, *, n)\mathcal{E}(z_n|z^{n-1})$
end for
2. Forward propagation of the weights
while $\mathrm{Dom}(w) \neq \{(\mathcal{E}, *, n+1) | \mathcal{E} \in \Xi, * \in \{u, f\}\}$ **do**
Pick $u \in \mathrm{Dom}(w)$ such that u is not the descendent of any $v \in \mathrm{Dom}(w)$.
for all v children of u **do**
if $v \notin \mathrm{Dom}(w)$ **then**
initalize $w(v) \leftarrow 0$.
end if
$w(v) \leftarrow w(v) + w(u)P(u \to v)$
end for
$\mathrm{Dom}(w) \leftarrow \mathrm{Dom}(w) \setminus \{u\}$
end while
Renormalize w.
3. Predict z_{n+1}: $P_{\text{Forward}}(z_{n+1}) = \sum_{\substack{\mathcal{E} \in \Xi \\ * \in \{u,f\}}} w(\mathcal{E}, *, n+1)\mathcal{E}(z_{n+1}|z^n)$

This algorithm actually computes π^*_{Switch} described in (4.26).

Theorem 4.17. *Let $\Xi = \{\mathcal{E}_1, ..., \mathcal{E}_n\}$ a set of experts, and let*

- $w_\Xi = \pi_K$
- *For all n, $\tau_n = \theta$*
- *For all n, $\sigma_n = \frac{\pi_\tau(n)}{\pi_\tau([n,+\infty[)}$*

Then Algorithm 1 for the switch HMM defined by w_Ξ, (τ_n) and σ_n computes the predictions from the switch distribution defined by π_M geometric with rate θ, π_K and π_τ. In other words:

$$\prod_{i=1}^{n} P_{\text{Forward}}(z_i|z^{<i}) = \sum_{\theta \in \Theta} \pi_{\text{Switch}}(\theta) \left[\phi(\theta)(z^n|\varnothing)\right], \tag{4.27}$$

where ϕ is from Definition 4.15 (in particular, $\phi(\theta)$ is an expert).

In particular, $\mathrm{Switch}_{\pi^*_{\text{Switch}}}(\mathcal{E}_1, ..., \mathcal{E}_n)(z^m|\varnothing)$ can be computed in $O(nmt)$, where t is the maximal computing time for the likelihoods $\mathcal{E}_k(z_i|z^{<i})$ (knowing the state of $\mathcal{E}_k$ before i).

See Appendix A.7 in [vEGdR12] (Theorem 14) for the proof.

If $\theta \in (0, 1)$, $\mathrm{Supp}(\pi_K) = \Xi$ and $\mathrm{Supp}(\pi_\tau) = n$, π^*_{Switch} has the intersting property that any sequence I that is constant for large n is given a positive

weight. Consequently, the weight of all the sequences of experts using for example $\mathcal{E}_1$ after the time n can be uniformly bounded by some $h = h(n) > 0$.

This property leads to a bound for the posterior weight of given sequences that is uniform on $\mathbf{z}$.

Lemma 4.18. *Let $\mathcal{E}_1$, ..., $\mathcal{E}_m$ be m experts from X to Y with same domain D and target T.*

Let μ be a probability distribution on $\{1, ..., m\}^{\mathbb{N}}$, and for all $I \in \{1, ..., n\}^{\mathbb{N}}$, $\mathbf{z} \in (X \times Y)^*$, let us write $\mu(I|\mathbf{z}) := \dfrac{\mu(I)\mathcal{E}_I(\mathbf{z}|\varnothing)}{\sum_{J\in\{1,...,n\}^{\mathbb{N}}} \mu(J)\mathcal{E}_J(\mathbf{z}|\varnothing)}$ the posterior weight after having seen $\mathbf{z}$ of the expert $\mathcal{E}_I$ in $\mathrm{Switch}_\mu(\mathcal{E}_1, ..., \mathcal{E}_m)$.

Let $n \in \mathbb{N}$, and let $I \in \{1, ..., m\}^{\mathbb{N}}$.

If there exists $h(n)$ such that for all $J \in \{1, ..., m\}^{\mathbb{N}}$, $J \sim_n I \implies \mu(J) \geqslant h(n)$, then, for all $\mathbf{z}$ such that $\phi(\mathbf{z}) := 1 + |\mathbf{z} \cap (D \times T)| \leqslant n$:

$$\max_{\substack{J\in\{1,...,m\}^{\mathbb{N}} \\ J\sim_{\phi(\mathbf{z})} I}} \mu(J|\mathbf{z}) \geqslant h(n). \tag{4.28}$$

6

Proof. The idea behind this proof is that one of the $\mathcal{E}_J$ for $J \sim_n I$ makes the best possible predictions, and consequently, its posterior weight will be greater than its prior weight. More precisely:

Let $\mathbf{z} = z^N \in (X \times Y)^N$.

For $1 \leqslant i \leqslant \phi(\mathbf{z}) - 1$, we write $d(i) = \inf\{j \in \mathbb{N} | \phi(z^j) = i + 1\}$ the index of the i-th point predicted by the $\mathcal{E}_k$, and $f_i(\mathbf{z}) := \mathrm{argmax}_k \mathcal{E}_k(z_{d(i)}|z^{d(i)-1})$ the index of the best expert at local time k. [7] Notice that by definition of the $d(i)$ the $\mathcal{E}_k$ issue a prediction for $z_{d(i)}$, i.e. $y_{d(i)} \in T$ and $(z^{d(i)-1}, x_{d(i)}) \in D$.

Let us now define the sequence J^* (depending on $\mathbf{z}$) by:

- $J^*_i = f_i(\mathbf{z})$ if $i < \phi(\mathbf{z})$
- $J^*_i = I_i$ if $i \geqslant \phi(\mathbf{z})$.

By definition, $J^* \sim_{\phi(\mathbf{z})} I$, so $J^* \sim_n I$. Moreover, for all $J \in \{1, ..., m\}^{\mathbb{N}}$, we have:

$$\mathcal{E}_J(\mathbf{z}|\varnothing) = \prod_{i=1}^{\phi(\mathbf{z})-1} \mathcal{E}_J(z_{d(i)}|z^{d(i)-1}) \tag{4.29}$$

$$\leqslant \prod_{i=1}^{\phi(\mathbf{z})-1} \mathcal{E}_{f_i(\mathbf{z})}(z_{d(i)}|z^{d(i)-1}) = \prod_{i=1}^{\phi(\mathbf{z})-1} \mathcal{E}_{J^*}(z_{d(i)}|z^{d(i)-1}) = \mathcal{E}_{J^*}(\mathbf{z}|\varnothing), \tag{4.30}$$

[6] We recall that $I \sim_n J \Leftrightarrow \forall k \geqslant n,\ I_k = J_k$.

[7] If the maximum is attained by several experts, choose the smallest index, for example.

since $\mathcal{E}_{J^*}(z_{d(i)}|z^{d(i)-1}) = \mathcal{E}_{J^*_{i_{\phi(z^{d(i)-1})}}}(z_{d(i)}|z^{d(i)-1}) = \mathcal{E}_{J^*_i}(z_{d(i)}|z^{d(i)-1})$.

Consequently, we have:

$$\mu(J^*|\mathbf{z}) = \frac{\mu(J^*)\mathcal{E}_{J^*}(\mathbf{z}|\varnothing)}{\sum_{J\in\{1,\dots,n\}^{\mathbb{N}}}\mu(J)\mathcal{E}_J(\mathbf{z}|\varnothing)} \tag{4.31}$$

$$\geqslant \frac{\mu(J^*)\mathcal{E}_{J^*}(\mathbf{z}|\varnothing)}{\sum_{J\in\{1,\dots,n\}^{\mathbb{N}}}\mu(J)\mathcal{E}_{J^*}(\mathbf{z}|\varnothing)} = \frac{\mu(J^*)}{\sum_{J\in\{1,\dots,n\}^{\mathbb{N}}}\mu(J)} = \mu(J^*) \geqslant h, \tag{4.32}$$

which is what we wanted. □

4.2.4 Restriction

This simple operation consists in restricting the domain of an expert to a smaller set:

Definition 4.19. *If $D' \subset D$, the restriction to D' of an expert $\mathcal{E}$ from X to Y with domain D is the expert $\mathcal{E}_{|D'}$ with domain D' defined by:*

$$\mathcal{E}_{|D'}(x, y|\mathbf{z}) = \mathcal{E}(x, y|\mathbf{z}) \tag{4.33}$$

for all $(\mathbf{z}, x) \in D'$, for all $y \in Y$.

4.2.5 Union

There are two different kinds of union for experts: a domain union (uniting two experts with disjoint domains), and a target union (uniting two experts with disjoint targets):

Suppose that we have two experts for regression: $\mathcal{E}_1$, an expert on $[0, 0.5)$, and $\mathcal{E}_2$, an expert on $[0.5, 1)$. It is straightforward to define an expert $\mathcal{E}_1 \cup \mathcal{E}_2$ on $[0, 1)$ that will use either $\mathcal{E}_1$ or $\mathcal{E}_2$, depending on where x falls. This is the domain union.

However, if we have two density estimation experts $\mathcal{E}_1$ and $\mathcal{E}_2$, respectively on $[0, 0.5)$ and on $[0.5, 1)$, we need a way to weight each of these experts in order to obtain a density on $[0, 1)$. This is the target union.

Defining the former is straightforward, but for the latter (which we want to use for density estimation), another expert has to be added: we need to choose if we will be using $\mathcal{E}_1$ or $\mathcal{E}_2$ for the next prediction.

4.2.5.1 Domain union

Given two different experts $\mathcal{E}_1$ and $\mathcal{E}_2$ with disjoint domains D_1 and D_2, it is straightforward to obtain a single expert with target $D_1 \cup D_2$ by using either $\mathcal{E}_1$ or $\mathcal{E}_2$, depending on where the data falls:

Definition 4.20. *If $\mathcal{E}_1, ...\mathcal{E}_n$ are experts with same target T and disjoint domains D_i, then the union of the $(\mathcal{E}_i)$, noted $\bigcup_{i=1}^n \mathcal{E}_i$, is the expert with domain $\bigcup_i D_i$ and target T defined by:*

$$(\bigcup_{i=1}^{n} \mathcal{E}_i)(y|\mathbf{z}, x) = \sum_{i=1}^{n} \mathcal{E}_i(z'|\mathbf{z}, x) 1_{(\mathbf{z},x)\in \mathrm{Dom}(\mathcal{E}_i)} \tag{4.34}$$

for $(\mathbf{z}, x) \in D$, $y \in T$.

An interesting property of the domain union is that if the experts do not peek out, it is essentialy a product:

Lemma 4.21. *Let $\mathcal{E}_1, ..., \mathcal{E}_m$ be m experts from X to Y with target T, with respective disjoint domains $D_1, ..., D_n$. We have:*

$$\bigcup_i \mathcal{E}_i(z^n|\mathbf{z}) = \prod_i \mathcal{E}_i(z^n|\mathbf{z}), \tag{4.35}$$

for any $\mathbf{z} \in (X \times Y)^*$, $z^n \in (X \times Y)^n$.

Proof. Let us write $z_k = (x_k, y_k)$ for all k, and $D = \bigcup_i D_i$. We have:

$$\bigcup_i \mathcal{E}_i(z^n|\mathbf{z}) = \prod_{\substack{l=1 \\ (\mathbf{z}\cdot z^{l-1}, y_l)\in D \\ y_l \in T}}^{n} \bigcup_i \mathcal{E}_i(z_l|\mathbf{z}\cdot z^{l-1}) \tag{4.36}$$

$$= \prod_{\substack{l=1 \\ (\mathbf{z}\cdot z^{l-1}, y_l)\in D \\ y_l \in T}}^{n} \sum_{i=1}^{m} \mathbf{1}_{(\mathbf{z}\cdot z^{l-1}, x)\in \mathrm{Dom}(\mathcal{E}_i)} \mathcal{E}_i(z_l|\mathbf{z}\cdot z^{l-1}). \tag{4.37}$$

Since the $\mathcal{E}_i$ have disjoint domains, all the sums above have exactly one term. Consequently, if we regroup all of these terms according to the expert used:

$$\bigcup_i \mathcal{E}_i(z^n|\mathbf{z}) = \prod_{i=1}^{m} \prod_{\substack{l=1 \\ \mathbf{z}\cdot z^{l-1}, x\in \mathrm{Dom}(\mathcal{E}_i) \\ y_l \in T}}^{n} \mathcal{E}_i(z_l|\mathbf{z}\cdot z^{l-1}) \tag{4.38}$$

$$= \prod_{i=1}^{m} \mathcal{E}_i(z^n|\mathbf{z}), \tag{4.39}$$

which is what we wanted. □

4.2.5.2 Target union

Given two density experts $\mathcal{E}_1$ and $\mathcal{E}_2$ with respective targets T_1 and T_2, we need some way to weight them to renormalize the density.

To do so, we introduce a particular class of experts.

Definition 4.22. *Let X, Y be two sets. If $\mathcal{E}$ is an expert from X to Y, and $Y = \{y_1, ..., y_n\}$ is finite, we call $\mathcal{E}$ a* choice expert. *If X is clear from the context, we will simply write "a choice expert on Y".*

From a formal point of view, this definition is not necessary: writing that an expert is a choice expert is the statement that it will be used in a union of experts (in particular, they are only used for generation, not for prediction).

We are now able to define the target union of two or more experts:

Definition 4.23. *If $\mathcal{E}_1, ...\mathcal{E}_n$ are experts with same domain D, and if $\mathcal{C}$ is a choice expert on $\{\mathcal{E}_1, ..., \mathcal{E}_n\}$, then the union of the $(\mathcal{E}_i)$ with choice expert $\mathcal{C}$, noted $\bigcup_{\mathcal{C},i=1}^{n} \mathcal{E}_i$ is the expert with domain D and target $\bigcup_i T_i$ defined by:*

$$(\bigcup_{\mathcal{C},i=1}^{n} \mathcal{E}_i)(y|\mathbf{z}, x) = \sum_{i=1}^{n} \mathcal{C}(\mathcal{E}_i|\mathbf{z}, x)\mathcal{E}_i(y|\mathbf{z}, x)1_{\mathbf{z},x\in T_i} \tag{4.40}$$

for $(\mathbf{z}, x) \in D$, $y \in T_1 \cup ... \cup T_n$.

The presence of an expert in the subscript of $\cup$ indicates if the union is a domain union or a target union: $\mathcal{E}_1 \cup \mathcal{E}_2$ is a domain union, and $\mathcal{E}_1 \cup_{\mathcal{C}} \mathcal{E}_2$ is a target union.

4.2.5.3 Properties

Although these two unions are different, they have similar properties.

We write them only for the target union (for the domain union, invert the target and the domain situations, and remove the choice experts).

By definition, $\cup$ is commutative. It is also associative in the following sense:

Proposition 4.24. *Let $\mathcal{E}_1$,$\mathcal{E}_2$,$\mathcal{E}_3$ be three X-experts, let $\mathcal{C}_1$ (resp. $\mathcal{C}'$) be a choice expert between $\mathcal{E}_1$ and $\mathcal{E}_2$ (resp. between $\mathcal{E}_1 \cup_{\mathcal{C}_1} \mathcal{E}_2$ and $\mathcal{E}_3$). Then:*

$$(\mathcal{E}_1 \cup_{\mathcal{C}_1} \mathcal{E}_2) \cup_{\mathcal{C}'} \mathcal{E}_3 = \bigcup_{\mathcal{C}} (\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3), \tag{4.41}$$

where $\mathcal{C}$ is the choice expert between $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$ defined by $\mathcal{C}(\mathcal{E}_1|\mathbf{z}, x) = \mathcal{C}'(\mathcal{E}_1 \cup_{\mathcal{C}_1} \mathcal{E}_2|\mathbf{z}, x)\mathcal{C}_1(\mathcal{E}_1|\mathbf{z}, x)$, $\mathcal{C}(\mathcal{E}_2|\mathbf{z}, x) = \mathcal{C}'(\mathcal{E}_1 \cup_{\mathcal{C}_1} \mathcal{E}_2|\mathbf{z}, x)\mathcal{C}_1(\mathcal{E}_2|\mathbf{z}, x)$ and $\mathcal{C}(\mathcal{E}_3|\mathbf{z}, x) = \mathcal{C}'(\mathcal{E}_3|\mathbf{z}, x)$.

Proof. Let $\mathbf{z}, x \in D$, $y \in T_1 \cup T_2 \cup T_3$. We have:

$$(\mathcal{E}_1 \cup_{\mathcal{C}_1} \mathcal{E}_2) \cup_{\mathcal{C}'} \mathcal{E}_3(y|\mathbf{z},x) = \mathcal{C}'(\mathcal{E}_1 \cup_{\mathcal{C}_1} \mathcal{E}_2|\mathbf{z},x)\mathcal{E}_1 \cup_{\mathcal{C}_1} \mathcal{E}_2(y|\mathbf{z},x)1_{(\mathbf{z},x)\in T_1\cup T_2} \tag{4.42}$$

$$+ \mathcal{C}'(\mathcal{E}_3|\mathbf{z},x)\mathcal{E}_3(y|\mathbf{z},x)1_{(\mathbf{z},x)\in T_3} \tag{4.43}$$

$$= \mathcal{C}'(\mathcal{E}_1 \cup_{\mathcal{C}_1} \mathcal{E}_2|\mathbf{z},x)\mathcal{C}'(\mathcal{E}_1|\mathbf{z},x)\mathcal{E}_1(y|\mathbf{z},x)1_{(\mathbf{z},x)\in T_1} \tag{4.44}$$

$$+ \mathcal{C}'(\mathcal{E}_1 \cup_{\mathcal{C}_1} \mathcal{E}_2|\mathbf{z},x)\mathcal{C}'(\mathcal{E}_2|\mathbf{z},x)\mathcal{E}_2(y|\mathbf{z},x)1_{(\mathbf{z},x)\in T_2} \tag{4.45}$$

$$+ \mathcal{C}'(\mathcal{E}_3|\mathbf{z},x)\mathcal{E}_3(y|\mathbf{z},x)1_{(\mathbf{z},x)\in T_3}, \tag{4.46}$$

which is what we wanted. □

There is a form of distributive property of the union over the Bayesian combination.

The following lemma is merely a technical tool:

Lemma 4.25. *If $\mathcal{E}_1$ and $\mathcal{E}_2$ are two experts with domain D and respective targets T_1 and T_2, with $T_1 \cap T_2 = \varnothing$, and if $\mathcal{C}$ is a choice expert between $\mathcal{E}_1$ and $\mathcal{E}_2$, then we have:*

$$(\mathcal{E}_1 \cup_{\mathcal{C}} \mathcal{E}_2)(\mathbf{z}'|\mathbf{z}) = \mathcal{E}_1(\mathbf{z}'|\mathbf{z})\mathcal{E}_2(\mathbf{z}'|\mathbf{z}) \prod_{\substack{1\leqslant i\leqslant n \\ \mathbf{z}\cdot\mathbf{z}'^{\leqslant i}\in D\times(T_1\cup T_2)}} \mathcal{C}(\arg(z_i)|\mathbf{z}\cdot\mathbf{z}'^{<i}), \tag{4.47}$$

with $\arg((x,y)) = \mathcal{E}_1$ *if* $y \in T_1$ *and* $\arg((x,y)) = \mathcal{E}_2$ *if* $y \in T_2$*, for any* $\mathbf{z}, \mathbf{z}' \in (X \times Y)^*$.

Proof. Let us write $\mathbf{z}' = ((x_1,y_1),(x_2,y_2),...)$.

$$\begin{aligned}
(\mathcal{E}_1 \cup_{\mathcal{C}} \mathcal{E}_2)(\mathbf{z}'|\mathbf{z}) =& \prod_{\substack{1\leqslant i\leqslant n \\ \mathbf{z}\cdot\mathbf{z}'^{\leqslant i}\in D\times(T_1\cup T_2)}} \left(\mathcal{C}(\mathcal{E}_1|\mathbf{z}\cdot\mathbf{z}'^{<i})\mathcal{E}_1(z_i'|\mathbf{z}\cdot\mathbf{z}'^{<i})1_{y_i\in T_1} + \mathcal{C}(\mathcal{E}_2|\mathbf{z}\cdot\mathbf{z}'^{<i})\mathcal{E}_2(z_i'|\mathbf{z}\cdot\mathbf{z}'^{<i})1_{y_i\in T_2}\right) \\
=& \prod_{\substack{1\leqslant i\leqslant n \\ \mathbf{z}\cdot\mathbf{z}'^{\leqslant i}\in D\times T_1}} \mathcal{C}(\mathcal{E}_1|\mathbf{z}\cdot\mathbf{z}'^{<i})\mathcal{E}_1(z_i'|\mathbf{z}\cdot\mathbf{z}'^{<i}) \prod_{\substack{1\leqslant i\leqslant n \\ \mathbf{z}\cdot\mathbf{z}'^{\leqslant i}\in D\times T_2}} \mathcal{C}(\mathcal{E}_2|\mathbf{z}\cdot\mathbf{z}'^{<i})\mathcal{E}_2(z_i'|\mathbf{z}\cdot\mathbf{z}'^{<i}) \\
=& \mathcal{E}_1(\mathbf{z}'|\mathbf{z})\mathcal{E}_2(\mathbf{z}'|\mathbf{z}) \prod_{\substack{1\leqslant i\leqslant n \\ \mathbf{z}\cdot\mathbf{z}'^{\leqslant i}\in D\times(T_1\cup T_2)}} \mathcal{C}(\arg(z_i)|\mathbf{z}\cdot\mathbf{z}'^{<i})
\end{aligned}$$

□

and the distributive property is the following:

Proposition 4.26 (Concatenation and Bayes)**.** *If $\mathcal{E}_i$ are experts with target T and domain D and $\mathcal{E}_i'$ are experts with target T' and domain D, with $T \cap T' = \varnothing$, and the w_i, w_i' are positive numbers, then for any expert $\mathcal{C}$:*

$$\mathrm{Bayes}_i((\mathcal{E}_i, w_i)) \cup_{\mathcal{C}} \mathrm{Bayes}_j((\mathcal{E}_j', w_j')) = \mathrm{Bayes}_{i,j}((\mathcal{E}_i \cup_C \mathcal{E}_j', w_i w_j')) \tag{4.48}$$

Proof. Let $\mathbf{z} = ((x_1, y_1), ..., (x_n, y_n))$, and let us write $\mathcal{B}_1 := \mathrm{Bayes}_i((\mathcal{E}_i, w_i))$ and $\mathcal{B}_2 := \mathrm{Bayes}_j((\mathcal{E}'_i, w'_i))$.

By lemma 4.25, we have:

$$\begin{aligned}\mathcal{B}_1 \cup_{\mathcal{C}} \mathcal{B}_2(\mathbf{z}|\varnothing) =& \mathcal{B}_1(\mathbf{z}|\varnothing)\mathcal{B}_2(\mathbf{z}|\varnothing) \prod_{\substack{1\leqslant k\leqslant n \\ z^k \in D\times(T\cup T')}} \mathcal{C}(\arg(z_k)|\mathbf{z}^{<k}) \\ =& \sum_i w_i \mathcal{E}_i(\mathbf{z}|\varnothing) \sum_j w'_j \mathcal{E}'_j(\mathbf{z}|\varnothing) \prod_{\substack{1\leqslant k\leqslant n \\ z^k \in D\times(T\cup T')}} \mathcal{C}(\arg(z_k)|\mathbf{z}^{<k}).\end{aligned}$$

and

$$\begin{aligned}\mathrm{Bayes}_{i,j}((\mathcal{E}_i \cup_C \mathcal{E}'_j, w_i w'_j))(\mathbf{z}|\varnothing) =& \sum_{i,j} w_i w'_j \left(\mathcal{E}_i \cup_C \mathcal{E}'_j\right)(\mathbf{z}|\varnothing) \\ =& \sum_{i,j} w_i w'_j \mathcal{E}_i(\mathbf{z}|\varnothing)\mathcal{E}'_j(\mathbf{z}|\varnothing) \prod_{\substack{1\leqslant k\leqslant n \\ z^k \in D\times(T\cup T')}} \mathcal{C}(\arg(z_k)|\mathbf{z}^{<k}).\end{aligned}$$

□

And as a consequence of the description of Switch as a Bayesian combination, this proposition extends to switch:

Corollary 4.27 (Concatenation and Switch)**.** *If $\mathcal{E}_i$ are experts with target T and domain D and $\mathcal{E}'_i$ are experts with target T' and domain D, with $T \cap T' = \varnothing$, and if μ and μ' are probability distributions on $\{1, ..., n\}^{\mathbb{N}}$, then for any expert $\mathcal{C}$ with domain D and target $\{\mathrm{Switch}_\mu((\mathcal{E}_i)), \mathrm{Switch}'_\mu((\mathcal{E}'_j))\}$:*

$$\mathrm{Switch}_\mu((\mathcal{E}_i)) \cup_{\mathcal{C}} \mathrm{Switch}'_\mu((\mathcal{E}'_j)) = \mathrm{Bayes}_{I,J}(\mathcal{E}_I \cup \mathcal{E}_J, \mu(I)\mu'(J)), \tag{4.49}$$

Proof. By Proposition 4.26,

$$\mathrm{Switch}_\mu((\mathcal{E}_i)) \cup_{\mathcal{C}} \mathrm{Switch}'_\mu((\mathcal{E}'_j)) = \mathrm{Bayes}_I(\mathcal{E}_I, \mu(I)) \cup_{\mathcal{C}} \mathrm{Bayes}_J(\mathcal{E}_J, \mu'(J)) \tag{4.50}$$

$$= \mathrm{Bayes}_{I,J}(\mathcal{E}_I \cup \mathcal{E}_J, \mu(I)\mu'(J)) \tag{4.51}$$

□

4.3 Expert trees

We can now describe the different algorithms that we will be using. The general idea is always the same: we build a tree such that each node s corresponds to an expert $\mathcal{E}_s$, deeper experts being more specialized, and, starting from the bottom of the tree we recursively choose if we are using the children of a node, or the node itself.

More precisely:

Definition 4.28. *Let X, Y be two sets. We call* expert tree *from X to Y a rooted tree T with root ε such that*

- *Each node s of T has a finite number of children. The set of the children of s is denoted by $c(s)$.*
- *Each node s contains an expert $\mathcal{E}_s$ from X to Y.*
- $\mathrm{Dom}(\mathcal{E}_s) = \bigcup_{t\in c(s)} \mathrm{Dom}(\mathcal{E}_t)$

We call weighted expert tree *an expert tree $\mathbf{T}$ with a probability distribution μ_s on $\{0,1\}^{\mathbb{N}}$ at each non-leaf node s.*

Intuitively, $\mathcal{E}_s$ is the local expert, and μ_s corresponds to the prior probability of opening the node s (i.e. with prior probability $\sum_{I,i_t=1} \mu_s(I)$, the "tree expert" at s uses the "tree experts" at the children of s instead of the local expert at s).

Notation 4.29. *If $\mathbf{T}$ is an expert tree, and if S is the set of leaves of $\mathbf{T}$, we denote by $\mathcal{E}_T$ the expert:*

$$\cup_{s\in S}\, \mathcal{E}_s \tag{4.52}$$

If $\mathcal{T} = (T_k)$ is a sequence of expert trees, we denote by $\mathcal{E}_{\mathcal{T}}$ the expert using $\mathcal{E}_{T_k}$ at time k.

As seen in Section 3.3, the CTW algorith recursively computes a Bayesian posterior at each node, finally yielding the prediction corresponding to the Bayesian combination of all the $\mathcal{E}_T$.

If we replace the Bayesian combination by a switching combination, we obtain another algorithm, Context Tree Switching [VNHB11], but it is not a switching combination of all the $\mathcal{E}_T$, because $\mathrm{Switch}\,(\mathcal{E}_1, \mathrm{Switch}(\mathcal{E}_2, \mathcal{E}_3))$ cannot be written $\mathrm{Switch}(\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3)$ in general. Consequently, we introduce *Switching patterns*, that can be though of as data-dependent trees. A switching pattern assigns to each node s a sequence in $I_s \in \{0,1\}^{\mathbb{N}}$ to each node, and $(I_s)_n$ indicates whether s is open (1) or closed (0) at time n. The prediction corresponding to the switching pattern is the prediction using the resulting tree.

Definition 4.30. *Let $\mathbf{T}$ be an expert tree. We call* switching pattern *on $\mathbf{T}$ an application matching each internal node s of $\mathbf{T}$ to $I_s \in \{0,1\}^{\mathbb{N}}$.*

If $\mathbf{T}$ is an expert tree from X to Y, and $\mathcal{I} = (I_s)$ a switching pattern on $\mathbf{T}$, for any node s of $\mathbf{T}$, we define the expert $\mathcal{E}_{\mathcal{I},s}$ by: $\mathrm{Dom}(\mathcal{E}_{\mathcal{I},s}) = \mathrm{Dom}(\mathcal{E}_s) =: D$, $\mathrm{Tar}(\mathcal{E}_{\mathcal{I},s}) = \mathrm{Tar}(\mathcal{E}_s) =: T$, *and*

- $\mathcal{E}_{\mathcal{I},s}(y|z^n, x) := \mathbf{1}_{(I_s)_k=0}\mathcal{E}_s(y|z^n, x) + \mathbf{1}_{(I_s)_k=1} \bigcup_{t\in c(s)} \mathcal{E}_{\mathcal{I},t}(y|z^n, x)$,

  for all $(z^n, x) \in D$, for all $y \in T$, where $k = k(s, z^n) := 1 + |\{z_k | 1 \leqslant k \leqslant n, (z^{<k}, x_k) \in D, y_k \in T\}|$, if s is not a leaf of $\mathbf{T}$.

- $\mathcal{E}_{\mathcal{I},s}(y|z^n,x) := \mathcal{E}_s(y|z^n,x)$ *otherwise.*

We define the weight $w_{\mathcal{I}}$ *of a switching pattern* $\mathcal{I}$ *as:*

$$w_{\mathcal{I}} = \prod_{\substack{s\in\mathbf{T}\\ s \text{ not a leaf of } \mathbf{T}}} \mu_s(I_s) \tag{4.53}$$

4.3.1 Context Tree Weighting

Context tree weighting has been introduced in [WST95] for text compression.

Let $\mathbf{T}$ be a finite weighted expert tree, such that for any internal node s of $\mathbf{T}$, $\mu_s(0^\infty) + \mu_s(1^\infty) = 1$. We write $w_s := \mu_s(0^\infty)$.

We denote by $\mathcal{T}_s$ the set of subtrees of $\mathbf{T}$ with root s such that each node t of the subtree has either 0 or $c(t)$ children.

We define the weight of $T \in \mathcal{T}_s$ as

$$w_T := \prod_{q \text{ internal nodes of } T} (1-w_s) \prod_{\substack{q \text{ leaves of } T\\ q \text{ not a leaf of } \mathbf{T}}} w_s \tag{4.54}$$

It can be checked that $\sum_{T\in\mathcal{T}_s} w_T = 1$.

Definition 4.31. *We define the* Context Tree Weighting *algorithm corresponding to the finite weighted expert tree* $\mathbf{T}$ *as:*

$$\mathrm{CTW}_{\mathbf{T}} := \mathrm{Bayes}_{T\in\mathcal{T}_\varepsilon}\left(\mathcal{E}_T, w_T\right) \tag{4.55}$$

The interesting point here is that this gigantic Bayesian combination can be computed "locally", starting from the bottom nodes, and going up to the root.

Theorem 4.32. *Let* $\mathbf{T}$ *be a weighted expert tree with* $w_s := \mu_s(0^\infty) = 1 - \mu_s(1^\infty)$*. For each node* s *of* $\mathbf{T}$*, we define the expert* CTW_s *as follows: The target of* CTW_s *is* $\mathrm{Tar}(\mathcal{E}_s)$*, its domain is* $\mathrm{Dom}(\mathcal{E}_s)$*, and*

$$\mathrm{CTW}_s = \mathrm{Bayes}\left((\mathcal{E}_s, w_s), \left(\bigcup_{\mathcal{C}_s, t\in c(s)} \mathrm{CTW}_t, 1-w_s\right)\right), \tag{4.56}$$

if s *is an internal node of* $\mathbf{T}$*, and*

$$\mathrm{CTW}_s = \mathcal{E}_s \tag{4.57}$$

if s *is a leaf of* $\mathbf{T}$*.*

With these notations, we have:

$$\mathrm{CTW}_s = \mathrm{Bayes}_{T\in\mathcal{T}_s}\left(\mathcal{E}_T, w_T\right), \tag{4.58}$$

or equivalently:

$$\mathrm{CTW}_s = \mathrm{Bayes}_{\mathcal{I}\,\text{switching pattern on}\,\mathbf{T}_s}\left(\mathcal{E}_{\mathcal{I},s}, w_{\mathcal{I}}\right), \tag{4.59}$$

where, if $\mathcal{I} = (I_s)$, $w_{\mathcal{I}} = \prod_{s\,\text{nodes of}\,\mathbf{T}} \mu_s(I_s)$.

In particular,

$$\mathrm{CTW}_\varepsilon = \mathrm{CTW}_{\mathbf{T}} \tag{4.60}$$

Proof. By induction on the nodes, starting from the leaves.

If s is a leaf, there is nothing to prove (the two experts are $\mathcal{E}_{E(s)}$).

Now, let s be an internal node such that the hypothesis is true for all descendents of s, and let $\{t_1, ..., t_n\} = c(s)$. By induction and lemma 4.26, we have:

$$\begin{aligned}
\bigcup_{i=1}^{n} \mathrm{CTW}_{t_i} =& \bigcup_{i=1}^{n} \mathrm{Bayes}_{T\in\mathcal{T}_{t_i}}\left(\mathcal{E}_T, w_T\right) \\
=& \mathrm{Bayes}_{\substack{T_1\in\mathcal{T}_{t_1} \\ \vdots \\ T_n\in\mathcal{T}_{t_n}}}\left(\mathcal{E}_{T_1} \cup ... \cup \mathcal{E}_{T_n}, w_{T_1}...w_{T_n}\right) \\
=& \mathrm{Bayes}_{\substack{T\in\mathcal{T}_s \\ T\neq\{s\}}}\left(\mathcal{E}_T, \frac{1}{1-w_s} w_T\right)
\end{aligned}$$

The theorem then easily follows from Corollary 4.11 and the fact that $\sum_{T\in\mathcal{T}_s, T\neq\{s\}} w_T = 1 - w_s$.

For the equivalence between (4.58) and (4.59), just remark that the only switching patterns $\mathcal{I}$ with non-zero weight are constant switching patterns, so $\mathcal{E}_{\mathcal{I},s} = \mathcal{E}_T$, where T is the subtree of $\mathbf{T}_s$ corresponding to the switching pattern $\mathcal{I}$, and $w_{\mathcal{I}} = w_T$, so the two Bayesian combinations are exactly the same. □

In the case of text compression, the original Context Tree Weighting algorithm uses Krichefsky–Trofimov estimators as local experts. Other experts can be used, as in [OHSS12], which uses "discounted" KT estimators (more weight on recent observations). Conditional NML [RR07] could also be a possibility.

It is also possible to recover the more recent Partition Tree Weighting algorithm [VWBG12] at fixed time horizon. We give more details about this in Section 4.3.4.3.

There are several more conditions for CTW to be implemented efficiently, they will be discussed in Section 4.3.4.

4.3.2 Context Tree Switching

[VNHB11] introduced a version of context tree switching, using a switch inspired by [vEGdR12]. We will define a more general algorithm, by replacing the Bayesian combination in context tree weighting (4.56) by a switch.

Definition 4.33. *Let* $\mathbf{T}$ *be a finite weighted expert tree. For each node* s *of* $\mathbf{T}$*, we define the expert* CTS_s *as follows: The target of* CTS_s *is* $\mathrm{Tar}(\mathcal{E}_s)$*, its domain is* $\mathrm{Dom}(\mathcal{E}_s)$*, and:*

$$\mathrm{CTS}_s = \mathrm{Switch}_{\mu_s}\left(\mathcal{E}_s, \bigcup_{t\in c(s)} \mathrm{CTS}_t\right), \tag{4.61}$$

if s *is an internal node of* $\mathbf{T}$*, and*

$$\mathrm{CTS}_s = \mathcal{E}_s \tag{4.62}$$

if s *is a leaf of* $\mathbf{T}$*.*

We write

$$\mathrm{CTS}_{\mathbf{T}} := \mathrm{CTS}_{\varepsilon}. \tag{4.63}$$

As for the context tree weighting algorithm, an unbounded depth context tree switching algorithm can be defined.

This algorithm does not generalize Definition 4.31 in the sense that Context Tree Switching will not actually be a switch combination of context trees. Actually, CTS is not even a Bayesian combination of switching patterns. However, it is still possible to obtain theoretical guarantees.

4.3.2.1 Properties

As an immediate consequence of the definition of the switch, we have:

Lemma 4.34. *Let* X*,* Y *be two sets, Let* μ *be a probability distribution on* $\{0,1\}^{\mathbb{N}}$*, let* $\mathcal{E}_0$*,* $\mathcal{E}_1$ *be two experts, and let* $\mathcal{E} = \mathrm{Switch}_{\mu}(\mathcal{E}_0, \mathcal{E}_1)$*. There exists an expert* $\mathcal{E}'$ *from* X *to* Y *such that*

$$\mathcal{E} = \mathrm{Bayes}((\mathcal{E}_0, k_0), (\mathcal{E}_1, k_1), (\mathcal{E}', 1-k_0-k_1)), \tag{4.64}$$

where $k_0 = \mu(0^{\infty})$ *and* $k_1 = \mu(1^{\infty})$*.*

Proof. (4.64) is exactly the definition of switch if

$$\mathcal{E}'(\mathbf{z}|\varnothing) = \frac{1}{1-k_0-k_1} \sum_{I\in\{0,1\}^{\mathbb{N}}\setminus\{(0^{\infty}),(1^{\infty})\}} \mu(I)\mathcal{E}_I(\mathbf{z}|\varnothing) \tag{4.65}$$

□

and therefore:

Corollary 4.35. *Let* $\theta \in \mathbb{R}$*, let* μ *be a probability distribution on* $\{0,1\}^{\mathbb{N}}$ *such that* $\min(\mu(0^{\infty}), \mu(1^{\infty})) \geqslant \frac{\theta}{2}$*, let* $\mathcal{E}_0$*,* $\mathcal{E}_1$ *be two experts, and let* $\mathcal{E} = \mathrm{Switch}_{\mu}(\mathcal{E}_0, \mathcal{E}_1)$*. We have:*

$$\mathcal{E}(\mathbf{z}|\varnothing) \geqslant \frac{\theta}{2}\left(\mathcal{E}_0(\mathbf{z}|\varnothing) + \mathcal{E}_1(\mathbf{z}|\varnothing)\right) \tag{4.66}$$

for all $\mathbf{z} \in (X\times Y)^*$

θ corresponds to the freezing probability at $t = 0$ in [vEGdR12] (hence the factor $\frac{1}{2}$.

Finally,

Theorem 4.36. *Let* $\mathbf{T}$ *be a finite weighted expert tree, let* s *be a node of* $\mathbf{T}$*, and let* $\mathcal{T}_s$ *be the set of subtrees of* $\mathbf{T}$ *with root* s*. There exists an expert* $\mathcal{E}'$ *on* $\mathrm{Tar}(\mathcal{E}_s)$ *such that*

$$\mathrm{CTS}_s = \mathrm{Bayes}\left((\mathcal{E}_T, w_s(T))_{T\in\mathcal{T}_s}, (\mathcal{E}', 1 - \sum w_s(T))\right) \tag{4.67}$$

where, if $\theta_s := \min(\mu_s(0^\infty), \mu_s(1^\infty))$*:*

$$w_s(T) = \prod_{\substack{q\in T \\ q \text{ not a leaf of } \mathbf{T}}} \theta_q, \tag{4.68}$$

In particular:

$$\mathrm{CTS}_s(\mathbf{z}|\varnothing) \geqslant \sum_{T\in\mathcal{T}_s} w(T)\mathcal{E}_T(\mathbf{z}|\varnothing), \tag{4.69}$$

Proof. We give the proof in the binary case, the generalization is straightforward. We denote the children of s by $s0$ and $s1$.

By induction on $|s|$. If $|s| = D$, then $\mathcal{T}_s = \{s\}$, and we do have $\mathrm{CTS}_s(x|\varnothing) = \mathcal{E}_s(x|\varnothing) \geqslant \frac{\theta_s}{2}\mathcal{E}_s(x|\varnothing)$, which is what we wanted.

Now, let s such that $0 \leqslant |s| \leqslant D-1$, and suppose that for all q longer than s, $\mathrm{CTS}_q(x|\varnothing) \geqslant \sum_{T\in\mathcal{T}_q} w(T)\mathcal{E}_T(x|\varnothing)$.

By Lemma 4.34, there exists $\mathcal{E}'$ such that:

$$\mathrm{CTS}_s = \mathrm{Switch}_s\left(\mathcal{E}_s, \mathrm{CTS}_{s0} \cup \mathrm{CTS}_{s1}\right) \tag{4.70}$$

$$= \mathrm{Bayes}\left((\mathcal{E}_s, \frac{\theta_s}{2}), (\mathrm{CTS}_{s0} \cup \mathrm{CTS}_{s1}, \frac{\theta_s}{2}), (\mathcal{E}', 1-\theta_s)\right) \tag{4.71}$$

Now, by induction, let $\mathcal{E}'_0$ and $\mathcal{E}'_1$ be such that:

$$\mathrm{CTS}_{s0} = \mathrm{Bayes}\left((\mathcal{E}_T, w_{s0}(T))_{T\in\mathcal{T}_{s0}}, (\mathcal{E}'_0, 1 - \sum w_{s0}(T))\right). \tag{4.72}$$

$$\mathrm{CTS}_{s1} = \mathrm{Bayes}\left((\mathcal{E}_T, w_{s1}(T))_{T\in\mathcal{T}_{s1}}, (\mathcal{E}'_1, 1 - \sum w_{s1}(T))\right). \tag{4.73}$$

We have, by Lemma 4.26:

$$\mathrm{CTS}_{s0}\cup\mathrm{CTS}_{s1} = \mathrm{Bayes}\left((\mathcal{E}_T \cup \mathcal{E}_{T'}, w_{s0}(T)w_{s1}(T'))_{\substack{T\in\mathcal{T}_{s0} \\ T'\in\mathcal{T}_{s1}}}, (\mathcal{E}'', 1 - \sum_{\substack{T\in\mathcal{T}_{s0} \\ T'\in\mathcal{T}_{s1}}} w_{s0}(T)w_{s1}(T'))\right), \tag{4.74}$$

where $\mathcal{E}''$ contains all the other terms.

Now, by combining equations (4.71) and (4.74), and using Lemma 4.26, we obtain:

$$\mathrm{CTS}_s = \mathrm{Bayes}\left((\mathcal{E}_s, \frac{\theta_s}{2}), \left(\mathcal{E}_T \cup \mathcal{E}_{T'}, \frac{\theta_s}{2} w_{s0}(T) w_{s1}(T')\right)_{\substack{T \in \mathcal{T}_{s0} \\ T' \in \mathcal{T}_{s1}}}, (\mathcal{E}''', 1 - \frac{\theta_s}{2} - \sum_{\substack{T \in \mathcal{T}_{s0} \\ T' \in \mathcal{T}_{s1}}} \frac{\theta_s}{2} w_{s0}(T) w_{s1}(T'))\right), \tag{4.75}$$

which is what we wanted. □

In particular, $\mathrm{CTS}_s(\mathbf{z}|\varnothing) \geqslant l\mathrm{CTW}_s(\mathbf{z}|\varnothing)$ if the product $\prod_{q \text{ all prefixes}} \theta_q$ converges to l, and we have $l > 0$ in any of the following cases:

- $\mathbf{T}$ is finite.
- $\min(\mu_s(0^\infty), \mu_s(1^\infty)) \geqslant \frac{1}{2}(1 - \frac{2^{-|s|}}{|s|^2})$. This is the case if the probability of starting unfrozen for a node at depth d is less than $\frac{2^{-d}}{d^2}$ (with prior opening weight $\frac{1}{2}$).
- We are given a sequence of positive numbers (u_t) such that $\sum u_t$ converges. Let N_t be the number of nodes opened at time t. The probability of starting unfrozen for a node s is then $a_{|s|} u_{\mathrm{ot}(s)}$, where (a_k) sums to one.

However, these conditions essentially amounts to preventing the algorithm from switching, which is not satisfying. In Chapter 5, we will show that in a particular case, CTS does behave better than CTW.

To do so, we will need the following lemma.

Lemma 4.37. *Let $\mathbf{T}$ be a (possibly infinite) weighted expert tree, and let $n_0 \in \mathbb{N}$. Let $\mathcal{I} := (\mathcal{I}_s)$ be a switching pattern on $\mathbf{T}$ such that for all s, $\mathcal{I}_s = 0^\infty$ or $\mathcal{I}_s \in \{0^k 1^\infty | 0 \leqslant k \leqslant n\}$ (i.e. the expert uses $\mathcal{E}_s$ for the first k points, and then switches to the children of s).*

For all $s \in \mathbf{T}$, we have:

$$\mathrm{CTS}_{\mathbf{T}_s}(\mathbf{z}|\mathbf{z}') \geqslant h(n_0)^{|(\mathbf{T}_s|\mathcal{I})|} \mathcal{E}_{\mathcal{I}|s}(\mathbf{z}|\mathbf{z}'), \tag{4.76}$$

for all $\mathbf{z}, \mathbf{z}' \in (X \times Y)^*$ such that $\phi(\mathbf{z}') := 1 + |\mathbf{z}' \cap \mathcal{E}_s| \leqslant n_0$,[8] where:

- *$h(n_0) := \inf\{\mu_s(J) | s \in \mathbf{T}, (J \sim_{n+1} 1^\infty \text{ or } J = 0^\infty)\}$,[9]*
- *$\mathcal{I}|s$ is the restriction of $\mathcal{I}$ to $\mathbf{T}_s$,*
- *$\mathbf{T}_s|\mathcal{I}$ is the tree obtained by removing the descendents of all the nodes t such that $\mathcal{I}_t = 0^\infty$ from $\mathbf{T}_s$.*

[8] We recall that by definition, $\mathcal{E}_I(.|\mathbf{z}) = \mathcal{E}_{i_{\phi(\mathbf{z})}}(.|\mathbf{z})$.

[9] We recall that $I \sim_n J \Leftrightarrow \forall k \geqslant n,\ I_k = J_k$.

Proof. If $\mathbf{T}|\mathcal{I}$ is infinite, then $h(n_0)^{|(\mathbf{T}_s|\mathcal{I})|} = 0$ and there is nothing to prove.[10] We now suppose that $\mathbf{T}|\mathcal{I}$ is finite, and we proceed by induction on s, starting with the leaves of $\mathbf{T}|\mathcal{I}$.

Preliminary computation.

Let $s \in \mathbf{T}|\mathcal{I}$, let $k \in [\![0, n_0]\!] \cup \{\infty\}$ such that $\mathcal{I}_s = (0^k 1^\infty)$.

Let us denote $T = \mathrm{Tar}(\mathcal{E}_s)$, $D = \mathrm{Dom}(\mathcal{E}_s)$, and let $t_1, ..., t_{|c(s)|}$ be an enumeration of the children of s. Let $\mathbf{z} \in (X \times Y)^*$ such that $\phi(\mathbf{z}) \leqslant n_0$. We have, for all n, for all $z^n = (z_1, ..., z_n) \in (X \times Y)^n$:

$$\mathrm{CTS}_s(z^n|\mathbf{z}) = \mathrm{Switch}_{\mu_s}(\mathcal{E}_s, \bigcup_{t \in c(s)} \mathrm{CTS}_t)(z^n|\mathbf{z}) \tag{4.77}$$

$$= \sum_{I_s \in \{0,1\}^{\mathbb{N}}} \mu_s(I_s|\mathbf{z}) \prod_{l=1}^{n} \left[\mathbf{1}_{I_{s_{\phi(\mathbf{z} \cdot z^l)}}=0} \mathcal{E}_s(y_l|\mathbf{z} \cdot z^{l-1}, x_l) + \mathbf{1}_{I_{s_{\phi(\mathbf{z} \cdot z^l)}}=1} \bigcup_{t \in c(s)} \mathrm{CTS}_t(y_l|\mathbf{z} \cdot z^{l-1}, x_l) \right]. \tag{4.78}$$

The first $\phi(\mathbf{z}) - 1$ terms of I_s are not used in the product above, so we regroup all the terms corresponding to sequences ending as $\mathcal{I}_s$:

$$\mathrm{CTS}_s(z^n|\mathbf{z}) \geqslant \sum_{\substack{I_s \in \{0,1\}^{\mathbb{N}} \\ I_s \sim_{\phi(\mathbf{z})} \mathcal{I}_s}} \mu_s(I_s|\mathbf{z}) \prod_{l=1}^{n} \left[\mathbf{1}_{\mathcal{I}_{s_{\phi(\mathbf{z} \cdot z^l)}}=0} \mathcal{E}_s(y_l|\mathbf{z} \cdot z^{l-1}, x_l) + \mathbf{1}_{\mathcal{I}_{s_{\phi(\mathbf{z} \cdot z^l)}}=1} \left(\bigcup_{t \in c(s)} \mathrm{CTS}_t \right)(y_l|\mathbf{z} \cdot z^{l-1}, x_l) \right] \tag{4.79}$$

$$\geqslant \max_{\substack{I_s \in \{0,1\}^{\mathbb{N}} \\ I_s \sim_{\phi(\mathbf{z})} \mathcal{I}_s}} \mu_s(I_s|\mathbf{z}) \prod_{l=1}^{n} \left[\mathbf{1}_{\phi(\mathbf{z} \cdot z^l) \leqslant k} \mathcal{E}_s(y_l|\mathbf{z} \cdot z^{l-1}, x_l) + \mathbf{1}_{\phi(\mathbf{z} \cdot z^l) > k} \left(\bigcup_{t \in c(s)} \mathrm{CTS}_t \right)(y_l|\mathbf{z} \cdot z^{l-1}, x_l) \right]. \tag{4.80}$$

Now, by Lemma 4.18, $\max_{\substack{I_s \in \{0,1\}^{\mathbb{N}} \\ I_s \sim_{\phi(\mathbf{z})} \mathcal{I}_s}} \mu_s(I_s|\mathbf{z}) \geqslant h(n_0)$, so:

$$\mathrm{CTS}_s(z^n|\mathbf{z}) \geqslant h(n_0) \prod_{l=1}^{n} \left[\mathbf{1}_{\phi(\mathbf{z} \cdot z^l) \leqslant k} \mathcal{E}_s(y_l|\mathbf{z} \cdot z^{l-1}, x_l) + \mathbf{1}_{\phi(\mathbf{z} \cdot z^l) > k} \left(\bigcup_{t \in c(s)} \mathrm{CTS}_t \right)(y_l|\mathbf{z} \cdot z^{l-1}, x_l) \right], \tag{4.81}$$

$$\geqslant h(n_0) \prod_{l=1}^{m} \left[\mathcal{E}_s(y_l|\mathbf{z} \cdot z^{l-1}, x_l) \right] \prod_{l=m+1}^{n} \left(\bigcup_{t \in c(s)} \mathrm{CTS}_t \right)(y_l|\mathbf{z} \cdot z^{l-1}, x_l) \tag{4.82}$$

[10] $h(n_0)$ cannot be equal to 1, since $\{J \in \{0,1\}^{\mathbb{N}} | \, (J \sim_{n+1} 1^\infty \text{ or } J = 0^\infty)\}$ contains at least two elements.

where $m := \max(\sup\{l | 1 \leqslant l \leqslant n, \phi(\mathbf{z} \cdot z^l) \leqslant k\}, 0)$

$$\geqslant h(n_0) \prod_{l=1}^{m} \left[\mathcal{E}_s(y_l|\mathbf{z} \cdot z^{l-1}, x_l)\right] \left(\bigcup_{t \in c(s)} \mathrm{CTS}_t\right)(z^{m+1:n}|\mathbf{z} \cdot z^m), \tag{4.83}$$

where we define $z^0 := \varnothing$ and $\prod_{l=1}^{0} f(l) := 1$, for any f. So finally, we have:

$$\mathrm{CTS}_s(z^n|\mathbf{z}) \geqslant h(n_0) \prod_{l=1}^{m} \left[\mathcal{E}_s(y_l|\mathbf{z} \cdot z^{l-1}, x_l)\right] \left(\prod_{t \in c(s)} \mathrm{CTS}_t(z^{m+1:n}|\mathbf{z} \cdot z^m)\right) \tag{4.84}$$

by lemma 4.21.

Actual induction.
If s is a leaf of $\mathbf{T}|\mathcal{I}$, then by definition, $k = \infty$, so we have $m = n$, and:

$$\mathrm{CTS}_s(z^n|\mathbf{z}) \geqslant h(n_0) \prod_{l=1}^{n} \left[\mathcal{E}_s(y_l|\mathbf{z} \cdot z^{l-1}, x_l)\right] \tag{4.85}$$

$$\geqslant h(n_0)\mathcal{E}_s(z^n|\mathbf{z}) = h(n_0)\mathcal{E}_{\mathcal{I}|s}(z^n|\mathbf{z}) \tag{4.86}$$

by definition of $\mathcal{E}_{\mathcal{I}|s}$. This is what we wanted.

If s is any non-leaf node (of $\mathbf{T}|\mathcal{I}$) such that (4.76) is true for all descendents of s, then $k \neq \infty$ by definition of $\mathbf{T}|\mathcal{I}$ (so $k \leqslant n_0$). Now, by defintion of m:

- either $\phi(\mathbf{z}) > k$, and then $m = 0$, so $z^m = 0$, and $\phi(\mathbf{z} \cdot z^m) = \phi(\mathbf{z}) \leqslant n_0$
- or $\phi(\mathbf{z}) \leqslant k$, and then, by definition of m, $\phi(\mathbf{z} \cdot z^m) \leqslant k \leqslant n_0$

In both cases $\phi(\mathbf{z} \cdot z^m) \leqslant n_0$, so by induction, and using the definition of $\mathcal{E}_{\mathcal{I}|s}$:

$$\mathrm{CTS}_s(z^n|\mathbf{z}) \geqslant h(n_0) \prod_{l=1}^{m} \left[\mathcal{E}_{\mathcal{I}|s}(y_l|\mathbf{z} \cdot z^{l-1}, x_l)\right] \left(\prod_{t \in c(s)} h(n_0)^{|(\mathbf{T}_t|\mathcal{I})|}\right) \prod_{t \in c(s)} \mathcal{E}_{\mathcal{I}|t}(z^{m+1:n}|\mathbf{z} \cdot z^m) \tag{4.87}$$

$$\geqslant h(n_0) \prod_{l=1}^{m} \left[\mathcal{E}_{\mathcal{I}|s}(y_l|\mathbf{z} \cdot z^{l-1}, x_l)\right] \left(\prod_{t \in c(s)} h(n_0)^{|(\mathbf{T}_t|\mathcal{I})|}\right) \left(\bigcup_{t \in c(s)} \mathcal{E}_{\mathcal{I}|t}\right)(z^{m+1:n}|\mathbf{z} \cdot z^m) \tag{4.88}$$

by Lemma 4.21 again, so finally:

$$\mathrm{CTS}_s(z^n|\mathbf{z}) \geqslant h(n_0)^{|(\mathbf{T}_s|\mathcal{I})|}\mathcal{E}_{\mathcal{I}|s}(z^m|\mathbf{z})\mathcal{E}_{\mathcal{I}|s}(z^{m+1:n}|\mathbf{z}\cdot z^m) \tag{4.89}$$

$$\geqslant h(n_0)^{|(\mathbf{T}_s|\mathcal{I})|}\mathcal{E}_{\mathcal{I}|s}(z^n|\mathbf{z}) \tag{4.90}$$

This is what we wanted.

□

4.3.3 Edgewise context tree algorithms

Since we cannot always control where our data falls, it is possible to have a lot of points on the left of the tree, and almost none on the right. If we open the root, we will predict better on the left, but we will not have enough points on the right, while if we close the root, we will not be able to use enough the information we have on the left. This problem can be solved if instead of opening or closing *nodes*, we open and close *vertices*. This leads us to the following definition:

We can now define the edgewise context tree switching algorithm[11]:

Definition 4.38. *We call* edgewise weighted expert tree *an expert tree* $\mathbf{T}$ *such that a probability distribution* $\mu_{s\to t}$ *on* $\{0,1\}^{\mathbb{N}}$ *is associated to each edge* $s \to t$*.*

Intuitively, $\mathcal{E}_s$ has the same role as in a weighted expert tree, but instead of controlling the status (open/closed) of a *node* s, $\mu_{s\to t}$ controls the status of the *vertex* $s \to t$.

The edgewise context tree switching algorithm can now be defined:

Definition 4.39. *Let* $\mathbf{T}$ *be a finite edgewise weighted expert tree. For each* $s \in \mathbf{T}$*, we define the edgewise context tree switching expert* ECTS_s *as follows: The target of* ECTS_s *is* $\mathrm{Tar}(\mathcal{E}_s)$*, its domain is* $\mathrm{Dom}(\mathcal{E}_s)$*, and:*

$$\mathrm{ECTS}_s = \bigcup_{t\in c(s)} \mathrm{Switch}_{\mu_{s\to t}}(\mathrm{ECTS}_t, (\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)}) \tag{4.91}$$

is s *is an internal node of* $\mathbf{T}$*, and*

$$\mathrm{ECTS}_s = \mathcal{E}_s \tag{4.92}$$

otherwise.

We write

$$\mathrm{ECTS}_{\mathbf{T}} := \mathrm{ECTS}_{\varepsilon}. \tag{4.93}$$

[11] We have seen that the context tree weighting algorithm is a frozen context tree switching algorithm (Section 4.3.2.1). Similarly, the edgewise context tree weighting algorithm is simply a frozen edgewise context tree switching, i.e. a context tree switching algorithm such that for all $s\mathbf{T}$ and $t \in c(s)$, $\mu_{s\to t}(0^\infty + 1^\infty = 1)$.

In the case that for all $s \in \mathbf{T}$ and $t \in c(s)$, $\mu_{s\to t}(0^\infty) + \mu_{s\to t}(1^\infty) = 1$, which is of particular interest, we will use the denomination "edgewise context tree weighting*" (*ECTW*).*

As for regular context tree switching, we will study this algorithm by using *edgewise* switching patterns, which are a straightforward generalization: instead of opening or closing nodes, we open or close edges, and we use the deepest expert with the right domain that can be attained with open edges. In other words, $I_{s\to t}$ controls whether we use the father $\mathcal{E}_s$ (0) or the child $\mathcal{E}_t$ (1):

Definition 4.40. *Let $\mathbf{T}$ be an expert tree. We call* edgewise switching pattern *on $\mathbf{T}$ an application $\mathcal{I}$ matching each edge $s \to t$ to $\mathcal{I}_{s\to t} \in \{0,1\}^{\mathbb{N}}$.*

If $\mathbf{T}$ is an expert tree from X to Y, and $\mathcal{I} = (\mathcal{I}_{s\to t})$ an edgewise switching pattern on $\mathbf{T}$, for any node s of $\mathbf{T}$, we define the expert $\mathcal{E}_{\mathcal{I},s}$ by: $\mathrm{Dom}(\mathcal{E}_{\mathcal{I},s}) = \mathrm{Dom}(\mathcal{E}_s) =: D$, $\mathrm{Tar}(\mathcal{E}_{\mathcal{I},s}) = \mathrm{Tar}(\mathcal{E}_s) =: T$, *and*

- $\mathcal{E}_{\mathcal{I},s}(y|\mathbf{z},x) := \left(\bigcup_{t\in c(s)} \mathcal{E}_{\mathcal{I}_{s\to t}}\right)(y|\mathbf{z},x)$

  for all $(\mathbf{z},x) \in D$, for all $y \in T$, where

  $\mathcal{E}_{\mathcal{I}_{s\to t}}(y|\mathbf{z},x) := \mathbf{1}_{(\mathcal{I}_{s\to t})_{\phi_t(\mathbf{z})}=0}(\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)}(y|\mathbf{z},x) + \mathbf{1}_{(\mathcal{I}_{s\to t})_{\phi_t(\mathbf{z})}=1}\mathcal{E}_{\mathcal{I},t}(y|\mathbf{z},x)$,

  with $\phi_t(z^n) := 1 + |\{z_k | 1 \leqslant k \leqslant n, (z^{<k}, x_k) \in \mathrm{Dom}(\mathcal{E}_t), y_k \in T\}|$, if s is not a leaf of $\mathbf{T}$.

- $\mathcal{E}_{\mathcal{I},s}(y|\mathbf{z},x) := \mathcal{E}_s(y|\mathbf{z},x)$ *otherwise.*

We will write $\mathcal{E}_{\mathcal{I}} := \mathcal{E}_{\mathcal{I},\varepsilon}$, where ε is the root of $\mathbf{T}$.
We define the weight $w_{\mathcal{I}}$ of a switching pattern $\mathcal{I}$ as:

$$w_{\mathcal{I}} = \prod_{\substack{s\in\mathbf{T}\\ t\in c(s)}} \mu_{s\to t}(\mathcal{I}_{s\to t}). \tag{4.94}$$

4.3.3.1 Edgewise Context Tree Weighting as a Bayesian combination

Edgewise context tree weighting behaves just as regular context tree weighting, except that now, a node can activate separately its children.

Theorem 4.41. *Let $\mathbf{T}$ be an edgewise weighted expert tree with $w_{s\to t} := \mu_{s\to t}(0^\infty) = 1 - \mu_{s\to t}(1^\infty)$. For each node s of $\mathbf{T}$, we define the edgewise context tree weighting expert* ECTW_s *as follows: The target of* ECTW_s *is* $\mathrm{Tar}(\mathcal{E}_s)$*, its domain is* $\mathrm{Dom}(\mathcal{E}_s)$*, and*

$$\mathrm{ECTW}_s = \bigcup_{t\in c(s)} \mathrm{Bayes}\left(((\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)}, w_{s\to t}), (\mathrm{ECTS}_t, 1 - w_{s\to t})\right), \tag{4.95}$$

if s is an internal node of $\mathbf{T}$, and

$$\mathrm{ECTW}_s = \mathcal{E}_s \tag{4.96}$$

if s is a leaf of $\mathbf{T}$.

With these notations, we have:

$$\mathrm{ECTW}_s = \mathrm{Bayes}_{\mathcal{I}\text{ edgewise switching pattern on }\mathbf{T}_s}\left(\mathcal{E}_{\mathcal{I},s}, w_{\mathcal{I}}\right), \tag{4.97}$$

where, if $\mathcal{I} = (\mathcal{I}_{s\to t})$, $w_{\mathcal{I}} = \prod_{\substack{s\in\mathbf{T}\\ t\in c(s)}} \mu_{s\to t}(\mathcal{I}_{s\to t})$.

Sketch of the proof. The general idea is the same as in Theorem 4.32, but the terms appearing in the equations are more complicated.

For simplicity, we suppose that $\mathbf{T}$ is a binary tree, and we proceed by induction on s starting with the leaves. If s is a leaf, there is nothing to prove.

Suppose s is an internal node of $\mathbf{T}$ such that (4.95) holds for all descendents of s, and let us write $c(s) = \{t_1, t_2\}$.

We have, by Lemma 4.26:

$$\mathrm{ECTW}_s = \bigcup_{t\in c(s)} \mathrm{Bayes}\left(((\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)}, w_{s\to t}), (\mathrm{ECTS}_t, 1-w_{s\to t})\right) \tag{4.98}$$

$$= \mathrm{Bayes}\Big[((\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_{t_1})} \cup (\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_{t_2})}, w_{s\to t_1} w_{s\to t_2}), \tag{4.99}$$

$$(\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_{t_1})} \cup \mathrm{ECTW}_{t_2}, w_{s\to t_1}(1-w_{s\to t_2}), \tag{4.100}$$

$$\mathrm{ECTW}_{t_1} \cup (\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_{t_2})}, (1-w_{s\to t_1}) w_{s\to t_2}, \tag{4.101}$$

$$+\,\mathrm{ECTW}_{t_1} \cup \mathrm{ECTW}_{t_2}, (1-w_{s\to t_1})(1-w_{s\to t_2})\Big] \tag{4.102}$$

And by induction, these four terms are Bayesian combinations corresponding respectively to switching patterns such that: $\mathcal{I}_{s\to t_1} = \mathcal{I}_{s\to t_2} = 0^\infty$, $\mathcal{I}_{s\to t_1} = 0^\infty$ and $\mathcal{I}_{s\to t_2} = 1^\infty$, $\mathcal{I}_{s\to t_1} = 1^\infty$ and $\mathcal{I}_{s\to t_2} = 0^\infty$, and $\mathcal{I}_{s\to t_1} = \mathcal{I}_{s\to t_2} = 1^\infty$, with their respective weights.

□

4.3.3.2 General properties of ECTS

We also have an analogous of Lemma 4.37 and its corollary, where the switching patterns are replaced by edgewise switching patterns:

Lemma 4.42. *Let $\mathbf{T}$ be a (possibly infinite) weighted expert tree, and let $n_0 \in \mathbb{N}$. Let $\mathcal{I} := (\mathcal{I}_{s\to t})$ be an edgewise switching pattern on $\mathbf{T}$ such that for all s, t, $\mathcal{I}_{s\to t} = 0^\infty$ or $\mathcal{I}_{s\to t} \in \{0^k 1^\infty | 0 \leqslant k \leqslant n_0\}$.*

For all $s \in \mathbf{T}|\mathcal{I}$ such that s is not a leaf of $\mathbf{T}|\mathcal{I}$,[12] *we have:*

$$\mathrm{ECTS}_{\mathbf{T}_s}(\mathbf{z}|\mathbf{z}') \geqslant h(n_0)^{n_{\mathrm{edge}}(\mathbf{T}_s|\mathcal{I})}\mathcal{E}_{\mathcal{I}|s}(\mathbf{z}|\mathbf{z}'), \tag{4.103}$$

for all $\mathbf{z}, \mathbf{z}' \in (X \times Y)^*$ such that $\phi_s(\mathbf{z}') := 1 + |\mathbf{z}' \cap \mathcal{E}_s| \leqslant n_0$,[13] *where:*

- $h(n_0) := \inf\{\mu_{s\to t}(J) | s \in \mathbf{T}, t \in c(s), (J \sim_{n_0+1} 1^\infty \text{ or } J = 0^\infty)\}$,[14]
- *$\mathcal{I}|s$ is the restriction of $\mathcal{I}$ to $\mathbf{T}_s$,*
- *$\mathbf{T}|\mathcal{I}$ is the tree obtained by removing the descendents of all the nodes t such that $\mathcal{I}_{s\to t} = 0^\infty$ from $\mathbf{T}$, where s is the father of t.*
- *$n_{\mathrm{edge}}(\mathbf{T}) := |\mathbf{T}| - 1$ is the number of edges of $\mathbf{T}$.*

Proof. The proof is essentially the same as the proof of Lemma 4.37:

We work by induction on $\mathcal{T}|\mathcal{I}$.

If $\mathbf{T}_s|\mathcal{I}$ is infinite, then $h(n_0)^{|(\mathbf{T}_s|\mathcal{I})|} = 0$ and there is nothing to prove.[15] We now suppose that $\mathbf{T}_s|\mathcal{I}$ is finite, and we proceed by induction on s, starting with the nodes of $\mathbf{T}_s|\mathcal{I}$ whose only children are leaves.

Preliminary computation.

Let s be a non-leaf node of $\mathbf{T}|\mathcal{I}$.

Let us denote $T = \mathrm{Tar}(\mathcal{E}_s)$, $D = \mathrm{Dom}(\mathcal{E}_s)$, and let $t_1, ..., t_{|c(s)|}$ be an enumeration of the children of s. Let $\mathbf{z} \in (X \times Y)^*$ such that $\phi_s(\mathbf{z}) \leqslant n_0$. We have, for all n, for all $z^n = (z_1, ..., z_n) \in (X \times Y)^n$:

$$\mathrm{ECTS}_s(z^n|\mathbf{z}) = \left(\bigcup_{t\in c(s)} \mathrm{Switch}_{\mu_{s\to t}}(\mathrm{ECTS}_t, (\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)})\right)(z^n|\mathbf{z}) \tag{4.104}$$

$$= \prod_{t\in c(s)} \mathrm{Switch}_{\mu_{s\to t}}(\mathrm{ECTS}_t, (\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)})(z^n|\mathbf{z}) =: \prod_{t\in c(s)} S_t \tag{4.105}$$

by Lemma 4.21.

Let $k = k(t) \in [\![0, n_0]\!] \cup \{\infty\}$ such that $\mathcal{I}_{s\to t} = (0^k 1^\infty)$. Now, by Definition 4.14, for all $t \in c(s)$, we have:

$$S_t = \sum_{I_{s\to t}\in\{0,1\}^{\mathbb{N}}} \mu_{s\to t}(I_{s\to t}|\mathbf{z}) \prod_{l=1}^{n} \left[\mathbf{1}_{I_{(s\to t)_{\phi(\mathbf{z}\cdot z^l)}}=0}(\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)}(y_l|\mathbf{z}\cdot z^{l-1}, x_l) + \mathbf{1}_{I_{(s\to t)_{\phi}(\mathbf{z}\cdot z^l)}=1}\mathrm{ECTS}_t(y_l|\mathbf{z}\cdot z^{l-1}, x_l)\right]. \tag{4.106}$$

[12] Technically, it is possible to take any $s \in \mathbf{T}$ but we are not interested in what happens elsewhere.

[13] We recall that by definition, $\mathcal{E}_I(.|\mathbf{z}) = \mathcal{E}_{i_{\phi(\mathbf{z})}}(.|\mathbf{z})$.

[14] We recall that $I \sim_n J \Leftrightarrow \forall k \geqslant n,\ I_k = J_k$.

[15] $h(n_0)$ cannot be equal to 1, since $\{J \in \{0,1\}^{\mathbb{N}} | (J \sim_{n+1} 1^\infty \text{ or } J = 0^\infty)\}$ contains at least two elements.

The first $\phi_t(\mathbf{z})-1$ terms of I_s are not used in the product above, so we regroup all the terms corresponding to sequences ending as $\mathcal{I}_s$:

$$S_t \geqslant \sum_{\substack{I_{s\to t}\in\{0,1\}^{\mathbb{N}}\\ I_{s\to t}\sim_{\phi_t(\mathbf{z})}\mathcal{I}_{s\to t}}} \mu_{s\to t}(I_{s\to t}|\mathbf{z})\prod_{l=1}^{n}\left[\mathbf{1}_{\mathcal{I}_{(s\to t)_{\phi_t(\mathbf{z}\cdot z^l)}}=0}(\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)}(y_l|\mathbf{z}\cdot z^{l-1},x_l)+\mathbf{1}_{\mathcal{I}_{(s\to t)_{\phi_t(\mathbf{z}\cdot z^l)}}=1}\mathrm{ECTS}_t(y_l|\mathbf{z}\cdot z^{l-1},x_l)\right] \tag{4.107}$$

$$\geqslant \left(\max_{\substack{I_{s\to t}\in\{0,1\}^{\mathbb{N}}\\ I_{s\to t}\sim_{\phi_t(\mathbf{z})}\mathcal{I}_{s\to t}}} \mu_{s\to t}(I_{s\to t}|\mathbf{z})\right)\prod_{l=1}^{n}\left[\mathbf{1}_{\mathcal{I}_{(s\to t)_{\phi_t(\mathbf{z}\cdot z^l)}}=0}(\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)}(y_l|\mathbf{z}\cdot z^{l-1},x_l)+\mathbf{1}_{\mathcal{I}_{(s\to t)_{\phi_t(\mathbf{z}\cdot z^l)}}=1}\mathrm{ECTS}_t(y_l|\mathbf{z}\cdot z^{l-1},x_l)\right] \tag{4.108}$$

Now, by Lemma 4.18, $\max_{\substack{I_{s\to t}\in\{0,1\}^{\mathbb{N}}\\ I_{s\to t}\sim_{\phi_t(\mathbf{z})}\mathcal{I}_{s\to t}}} \mu_{s\to t}(I_{s\to t}|\mathbf{z})\geqslant h(n_0)$, so:

$$S_t \geqslant h(n_0)\prod_{l=1}^{n}\left[\mathbf{1}_{\mathcal{I}_{s\to t_{\phi_t(\mathbf{z}\cdot z^l)}}=0}(\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)}(y_l|\mathbf{z}\cdot z^{l-1},x_l)+\mathbf{1}_{\mathcal{I}_{s\to t_{\phi_t(\mathbf{z}\cdot z^l)}}=1}\mathrm{ECTS}_t(y_l|\mathbf{z}\cdot z^{l-1},x_l)\right], \tag{4.109}$$

$$\geqslant h(n_0)\prod_{l=1}^{m}\left[(\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)}(y_l|\mathbf{z}\cdot z^{l-1},x_l)\right]\prod_{l=m+1}^{n}\mathrm{ECTS}_t(y_l|\mathbf{z}\cdot z^{l-1},x_l) \tag{4.110}$$

where $m=m(t):=\max(0,\sup\{l|1\leqslant l\leqslant n,\phi_t(\mathbf{z}\cdot z^l)\leqslant k\})$, since by definition of the $\mathcal{I}_{s\to t}$, $\mathcal{I}_{s\to t_{\phi_t(\mathbf{z}\cdot z^l)}}=0$ iff $l\leqslant m$.

$$\geqslant h(n_0)\left[(\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)}(z^m|\mathbf{z})\right]\mathrm{ECTS}_t(z^{m+1:n}|\mathbf{z}\cdot z^m), \tag{4.111}$$

Where we define $z^0:=\varnothing$ and $\prod_{l=1}^{0}f(l):=1$, for any f.

In particular, if $\mathcal{I}_{s\to t}=0^\infty$, we have: $k(t)=\infty$, so $m=n$, and:

$$S_t=\mathrm{Switch}_{\mu_{s\to t}}(\mathrm{ECTS}_t,(\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)})(z^n|\mathbf{z})\geqslant h(n_0)(\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)}(z^n|\mathbf{z})=h(n_0)\mathcal{E}_{\mathcal{I}_{s\to t}}(z^n|\mathbf{z}), \tag{4.112}$$

by definition of $\mathcal{E}_{\mathcal{I}_{s\to t}}$.

Actual induction.

If all the children of s in $\mathbf{T}_s|\mathcal{I}$ are leaves (i.e. $\mathcal{I}_{s\to t}=0^\infty$ for all $t\in c(s)$),

then by (4.112),

$$\mathrm{ECTS}_s(z^n|\mathbf{z}) = \prod_{t\in c(s)} S_t \geqslant h(n_0)^{|c(s)|} \prod_{t\in c(s)} (\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)}(z^n|\mathbf{z}) \tag{4.113}$$

$$\geqslant h(n_0)^{n_{\mathrm{edge}}(\mathbf{T}_s|\mathcal{I})} \left(\bigcup_{t\in c(s)} (\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)} \right)(z^n|\mathbf{z}) \tag{4.114}$$

by lemma 4.21,

$$\geqslant h(n_0)^{n_{\mathrm{edge}}(\mathbf{T}_s|\mathcal{I})} \mathcal{E}_s(z^n|\mathbf{z}) = h(n_0)^{n_{\mathrm{edge}}(\mathbf{T}_s|\mathcal{I})} \mathcal{E}_{\mathcal{I}|s}(z^n|\mathbf{z}) \tag{4.115}$$

and (4.103) holds.

Now, if s is any non-leaf node (of $\mathbf{T}|\mathcal{I}$) such that (4.103) is true for all the children t of s satisfying $\mathcal{I}_{s\to t} \neq 0^\infty$, we have, from equation (4.105):

$$\mathrm{ECTS}_s(z^n|\mathbf{z}) = \prod_{\substack{t\in c(s)\\ \mathcal{I}_{s\to t}\neq 0^\infty}} S_t \prod_{\substack{t\in c(s)\\ \mathcal{I}_{s\to t}= 0^\infty}} S_t. \tag{4.116}$$

If t is such that $\mathcal{I}_{s\to t} \neq 0^\infty$, then $k(t) \neq \infty$ by definition of $\mathbf{T}|\mathcal{I}$ (so $k(t) \leqslant n_0$). So, by definition of m, $\phi_t(\mathbf{z}\cdot z^m) \leqslant k(t) \leqslant n_0$

Consequently, by induction, using (4.111):

$$S_t \geqslant h(n_0) \left[(\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)}(z^m|\mathbf{z})\right] \mathrm{ECTS}_t(z^{m+1:n}|\mathbf{z}\cdot z^m) \tag{4.117}$$

$$\geqslant h(n_0) \left[(\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)}(z^m|\mathbf{z})\right] h(n_0)^{n_{\mathrm{edge}}(\mathbf{T}_t|\mathcal{I})} \mathcal{E}_{\mathcal{I}|t}(z^{m+1:n}|\mathbf{z}\cdot z^m) \tag{4.118}$$

$$\geqslant h(n_0)^{n_{\mathrm{edge}}(\mathbf{T}_t|\mathcal{I})+1} (\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)}(z^m|\mathbf{z}) \mathcal{E}_{\mathcal{I}|t}(z^{m+1:n}|\mathbf{z}\cdot z^m) \tag{4.119}$$

$$\geqslant h(n_0)^{n_{\mathrm{edge}}(\mathbf{T}_t|\mathcal{I})+1} \mathcal{E}_{\mathcal{I}_{s\to t}}(z^m|\mathbf{z}) \mathcal{E}_{\mathcal{I}_{s\to t}}(z^{m+1:n}|\mathbf{z}\cdot z^m) = h(n_0)^{n_{\mathrm{edge}}(\mathbf{T}_t|\mathcal{I})+1} \mathcal{E}_{\mathcal{I}_{s\to t}}(z^n|\mathbf{z}), \tag{4.120}$$

by definition of $\mathcal{E}_{\mathcal{I}_{s\to t}}$.

So by injecting (4.112) and (4.120) in (4.116):

$$\mathrm{ECTS}_s(z^n|\mathbf{z}) = \prod_{\substack{t\in c(s)\\ \mathcal{I}_{s\to t}\neq 0^\infty}} S_t \prod_{\substack{t\in c(s)\\ \mathcal{I}_{s\to t}= 0^\infty}} S_t \tag{4.121}$$

$$\geqslant \prod_{\substack{t\in c(s)\\ \mathcal{I}_{s\to t}\neq 0^\infty}} \left(h(n_0)^{n_{\mathrm{edge}}(\mathbf{T}_t|\mathcal{I})+1} \mathcal{E}_{\mathcal{I}_{s\to t}}(z^n|\mathbf{z})\right) \prod_{\substack{t\in c(s)\\ \mathcal{I}_{s\to t}= 0^\infty}} (h(n_0)\mathcal{E}_{\mathcal{I}_{s\to t}}(z^n|\mathbf{z})) \tag{4.122}$$

and since $n_{\mathrm{edge}}(\mathbf{T}_s|\mathcal{I}) = |c(s)| + \sum_{\substack{t\in c(s)\\ \mathcal{I}_{s\to t}\neq 0^\infty}} n_{\mathrm{edge}}(\mathbf{T}_t|\mathcal{I})$,

$$\mathrm{ECTS}_s(z^n|\mathbf{z}) \geqslant h(n_0)^{|c(s)|+\sum_{\substack{t\in c(s)\\ \mathcal{I}_{s\to t}\neq 0^\infty}} n_{\mathrm{edge}}(\mathbf{T}_t|\mathcal{I})} \prod_{t\in c(s)} \mathcal{E}_{\mathcal{I}_{s\to t}}(z^n|\mathbf{z}) = h(n_0)^{n_{\mathrm{edge}}(\mathbf{T}_s|\mathcal{I})} \prod_{t\in c(s)} \mathcal{E}_{\mathcal{I}_{s\to t}}(z^n|\mathbf{z}). \tag{4.123}$$

We conclude with Lemma 4.21 and the definition of $\mathcal{E}_{\mathcal{I}|s}(z^n|\mathbf{z}) = \bigcup_{t\in c(s)} \mathcal{E}_{\mathcal{I}_{s\to t}}(z^n|\mathbf{z})$. □

4.3.4 Practical use

For all these expert tree algorithms to be implemented properly, it is necessary to make sure that new data points only affects the experts in *one* branch of the tree: this is exactly the no peeking-out condition.

We only defined our algorithms on finite expert trees. As for the original CTW algorithm, it is possible to define unbounded depth algorithms on expert trees if the union involved at the nodes of the tree are domain unions. We do this in the following section.

4.3.4.1 Infinite depth algorithms

For an infinite depth CTW[16] algorithm to be defined, we need that for all $\mathbf{z} \in (X \times Y)^*$, there exists a weighted expert tree $\mathbf{T}$ such that for all $\mathbf{T}'$ such that $\mathbf{T}$ is a subtree of $\mathbf{T}$, $\mathrm{CTS}_{\mathbf{T}}(\mathbf{z}|\varnothing) = \mathrm{CTS}_{\mathbf{T}'}(\mathbf{z}|\varnothing)$.

This condition is satisfied if the experts in $\mathbf{T}$ are compatible.

Moreover, we want to prevent two different points from corresponding to the same infinite branch of the tree.

This leads us to the following definition:

Definition 4.43. *We say that an expert tree* $\mathbf{T}$ *is a* proper *expert tree if:*

- *The experts in* $\mathbf{T}$ *are compatible.*
- *The experts in* $\mathbf{T}$ *do not peek out.*
- $\lim_{k\to\infty} \sup\{|\mathrm{Dom}(\mathcal{E}_s)|, |s| = k\} = 0$ *(i.e. the domains shrink when depth increases)*[17]

Now, suppose we are trying to compute $\mathrm{CTW}_{\mathbf{T}}(y_n|z^{n-1}, x_n)$ under the following conditions:

- $\mathbf{T}$ is a full expert tree with depth D "large".
- $\mathbf{T}$ is proper.
- The x_i are pairwise distinct.
- We define s_0 as the first node (in terms of depth) satisfying $x_n \in \mathrm{Dom}(\mathcal{E}_{s_0})$ and $\forall k < n, x_k \notin \mathrm{Dom}(\mathcal{E}_{s_0})$ ("D large" in the first condition means "s_0 exists").

[16] We only describe the CTW algorithm here for simplicity, but the remarks in this section are also valid for the other expert tree algorithms.

[17] This condition is mostly useful for regression and density estimation. It can be applied to text compression with alphabet A by setting $\{x \in A^* | x = x^n; x^{n-|s|+1:n} = s\} := |A|^{|-s|}$.

In that case have $\mathrm{CTW}_{s_0}(z_n|z^{n-1}) = \mathrm{CTW}_{s_0}(z_n|\varnothing)$ (no peeking out), and since the experts in $\mathbf{T}$ are compatible, $\mathrm{CTW}_{s_0}(z_n|\varnothing) = \mathcal{E}_{s_0}(z_n|\varnothing)$ (if nothing has been seen in the context s yet, then a simple proof by induction yields $\mathrm{CTW}_s(\varnothing) = \mathcal{E}_s(\varnothing)$). In particular, $\mathrm{CTW}_{s_0}(z_n|z^{n-1})$ does not depend on D for D large: the only nodes of the tree for which computations are needed are the nodes that have already been visited at least once.

Notice however that for practical implementation of the infinite depth algorithm, it is necessary to store z_n at the node s_0. If another point z_m with $m > n$ is such that $z_m \in \mathrm{Dom}(\mathcal{E}_{s_0})$, then z_n has to be moved to the first node not also used for z_m. The algorithm is described in the Appendix.

Definition 4.44. *Let $\mathbf{T}$ be a (not necessarily finite) proper weighted expert tree.*

We define the Context Tree Weighting expert corresponding to $\mathbf{T}$ by: $\mathrm{Tar}(\mathrm{CTW}_{\mathbf{T}}) = \mathrm{Tar}(\mathrm{CTW}_{\mathbf{T}_0})$, $\mathrm{Dom}(\mathrm{CTW}_{\mathbf{T}}) = \mathrm{Dom}(\mathrm{CTW}_{\mathbf{T}_0}) \cap \mathcal{D}$, *and:*

$$\mathrm{CTW}_{\mathbf{T}}(z_n|z^{n-1}) = \lim_{D\to\infty} \mathrm{CTW}_{\mathbf{T}_D}(z_n|z^{n-1}), \tag{4.124}$$

where $\mathbf{T}_D$ is the subtree of $\mathbf{T}$ obtained by removing all nodes at depth $> D$, and $\mathcal{D} = \{(x_1, y_1), ..., (x_n, y_n), x_{n+1} \in (X \times Y)^* \times X | \forall k, k' \in \{1, ..., n+1\}, \exists s \in \mathbf{T}, (x_k \in \mathrm{Dom}(\mathcal{E}_s) \,\mathrm{and}\, x_{k'} \notin \mathrm{Dom}(\mathcal{E}_s))\}$.

In the infinite case, we restrict the domain to make sure that the Bayesian posteriors can still be normalized. Indeed, $(z^n, x_{n+1}) \in \mathcal{D}$ ensures that only a finite number of experts will have to be considered to compute the prediction $\mathrm{CTW}_{\mathbf{T}}(y_{n+1}|z_n, x_{n+1})$. On $[0, 1]$, if $\mathbf{T}$ is proper and if the x_i are i.i.d following a probability distribution with bounded density, $x^n \in \mathcal{D}$ with probability one, because of the condition $\lim_{k\to\infty} \sup\{|\mathrm{Dom}(\mathcal{E}_s)|, |s| = k\} = 0$.

4.3.4.1.1 Properties Lemmas 4.37 and 4.42, stating that the Context Tree Switching algorithms do "better" than given sequences of trees can be generalized to the infinite case. More precisely, we have:

Lemma 4.45. *Let $\mathbf{T}$ be a proper infinite weighted expert tree. Let $\mathcal{I} := (\mathcal{I}_s)$ be a switching pattern on $\mathbf{T}$ such that:*

- *for all s, $\mathcal{I}_s \in \{0^\infty, 01^\infty, 001^\infty\}$.*
- *There exists $D \in \mathbb{N}$ such that for all s such that $|s| = D$, $\mathcal{I}_s = 0^\infty$.*

For all $s \in \mathbf{T}$, we have:

$$\mathrm{CTS}_{\mathbf{T}_s}(\mathbf{z}|\varnothing) \geqslant h^{2^{D+1}-1} \mathcal{E}_{\mathcal{I}|s}(\mathbf{z}|\varnothing), \tag{4.125}$$

for all $\mathbf{z} \in (X \times Y)^*$, where $\mathcal{I}|s$ is the restriction of $\mathcal{I}$ to $\mathbf{T}_s$, and $h = \inf\{\mu_t(J) | t \in \mathbf{T}, (J \sim_3 1^\infty \,\mathrm{or}\, J = 0^\infty)\}$.

Proof. Direct application of Lemma 4.37 with $n_0 = 2$, noting that a binary tree with maximal depth D has $2^{D+1} - 1$ nodes. □

Lemma 4.46. *Let* $\mathbf{T}$ *be a proper infinite edgewise weighted expert tree. Let* $\mathcal{I} := (\mathcal{I}_{s\to t})$ *be a switching pattern on* $\mathbf{T}$ *such that:*

- *for all* s, $\mathcal{I}_{s\to t} \in \{0^\infty, 01^\infty, 001^\infty\}$.
- *There exists* $D \in \mathbb{N}$ *such that for all* s *such that* $|s| = D$, *for all* $t \in c(s)$, $\mathcal{I}_{s\to t} = 0^\infty$.

For all $s \in \mathbf{T}$, *we have:*

$$\mathrm{ECTS}_{\mathbf{T}_s}(\mathbf{z}|\varnothing) \geqslant h^{2^{D-|s|+2}-2}\mathcal{E}_{\mathcal{I}|s}(\mathbf{z}|\varnothing), \tag{4.126}$$

for all $\mathbf{z} \in (X \times Y)^*$, *where* $\mathcal{I}|s$ *is the restriction of* $\mathcal{I}$ *to* $\mathbf{T}_s$, *and* $h = \inf\{\mu_{t\to t'}(J)|t \in \mathbf{T}, t' \in c(t), (J \sim_3 1^\infty \text{ or } J = 0^\infty)\}$.

Proof. Direct application of Lemma 4.42 with $n_0 = 2$, noting that a binary tree with maximal depth D has $2^{D+1} - 2$ edges, and that the depth of $\mathbf{T}_s|\mathcal{I}$ is $D + 1 - |s|$. □

We now give some example of experts that could be used for expert tree algorithms in different situations.

4.3.4.2 Density estimation

The algorithms above have been described only for experts that can work with domain union. It is not the case for density estimation experts, which need target union. The algorithms can be adapted by adding at each node s a choice expert $\mathcal{C}_s$ for the union at s.

It is however harder to define an unbounded CTW algorithm for density estimation, since contrarily to the domain union case, the sequence $(\mathrm{CTW}_{\mathbf{T}_D})_{D\in\mathbb{N}}$ is not constant at infinity.

To give more intuition on Context Tree Weighting applied to density estimation, we give an example with one point:

Logarithmic singularity with only one point. Consider the following weighted expert tree for density on $[0, 1]$: each node s of $\mathbf{T}$ has two children: $s0$ and $s1$, for each s, $\mathcal{E}_s$ is the uniform density expert with target the set of numbers whose binary expansion start with s, $w_s = \frac{1}{2}$ for all s, and the choice experts $\mathcal{C}_s$ are defined by: $\mathcal{C}_s(\mathcal{E}_{s0}|x^n) = \frac{|\{k\in\{1,\dots,n\}|x_k\in\mathrm{Dom}(\mathcal{E}_{s0})\}|}{|\{k\in\{1,\dots,n\}|x_k\in\mathrm{Dom}(\mathcal{E}_s)\}|}$ if $\{k \in \{1, ..., n\}|x_k \in \mathrm{Dom}(\mathcal{E}_s)\} \neq \varnothing$, and $\mathcal{C}_s(\mathcal{E}_{s0}|x^n) = \frac{1}{2}$ otherwise.

Then we have:

Proposition 4.47. *The density* ϕ_{x_0} *output by the* $\mathrm{CTW}_{\mathbf{T}}$ *algorithm with* $\mathbf{T}$ *described above after having observed one single point* $x_0 = \sum_{i\geqslant 1} a_i 2^{-i}$ *(with* $a_i = 0$ *or* 1*) satisfies:*

$$\phi_{x_0}(\sum b_i 2^{-i}) = \frac{1}{2}\inf\{i, a_i \neq b_i\}, \tag{4.127}$$

where $b_i = 0$ *or* 1.

In particular, we have $\phi_0(x) = -\lfloor log_2(x) \rfloor$.

Proof. By definition of $\mathcal{E}_s$, we have $x_0 \in \mathrm{Tar}(\mathcal{E}_s)$ iff s is a prefix of $(a_1, a_2, ...)$.

Now, let $x := \sum_{i \geqslant 1} b_i 2^{-i} \in [0,1] \setminus \{x_0\}$.

$$\mathrm{CTW}_{\mathbf{T}}(x|x_0) = \mathrm{Bayes}_T(\mathcal{E}_T, w_T)(x|x_0) \tag{4.128}$$

$$= \sum_T \frac{w_T \mathcal{E}_T(x_0|\varnothing)}{\sum_T w_T \mathcal{E}_T(x_0|\varnothing)} \mathcal{E}_T(x|x_0) \tag{4.129}$$

$$= \sum_T w_T \mathcal{E}_T(x|x_0), \tag{4.130}$$

since $\mathcal{E}_T(x_0|\varnothing) = 1$ and thus $\sum_T w_T \mathcal{E}_T(x_0|\varnothing) = \sum_T w_T = 1$.

We now regroup the trees according to the leaf containing x_0. Let us denote by $f_{x_0}(T)$ the leaf of T such that $x_0 \in \mathrm{Dom}(\mathcal{E}_{f_{x_0}(T)})$:

$$\mathrm{CTW}_{\mathbf{T}}(x|x_0) = \sum_{n=0}^{\infty} \sum_{\substack{T \\ f_{x_0}(T) = a_1, ..., a_n}} w_T \mathcal{E}_T(x|x_0). \tag{4.131}$$

If $f_{x_0}(T) = a_1, ..., a_n$, then $\mathcal{E}_T(x|x_0) = 2^n \mathbf{1}_{\inf\{i, a_i \neq b_i\} > n}$. Indeed, the choice expert ensures that $\mathcal{E}_T(x|x_0) = \mathcal{E}_{f_{x_0}(T)}(x|x_0)$ if $x \in \mathrm{Dom}(\mathcal{E}_{f_{x_0}(T)})$ and $\mathcal{E}_T(x|x_0) = 0$ otherwise.

Moreover, $\mathcal{E}_T(x_0|\varnothing) = 1$ for all T. So we have:

$$CTW_{\mathbf{T}}(x|x_0) = \sum_{n=0}^{\infty} 2^n \mathbf{1}_{\inf\{i, a_i \neq b_i\} > n} \sum_{\substack{T \\ f_{x_0}(T) = a_1, ..., a_n}} w_T. \tag{4.132}$$

But for $n \in \mathbb{N}$, $\sum_{\substack{T \\ f_{x_0}(T) = a_1, ..., a_n}} w_T = \frac{1}{2^{n+1}}$, since $\{T | f_{x_0}(T) = a_1, ..., a_n\}$ can be characterized exactly by " the nodes ε;a_1; a_1, a_2;...; $a_1, ..., a_{n-1}$ are open, and $a_1, ..., a_n$ is closed".

So finally,

$$\mathrm{CTW}_{\mathbf{T}}(x|x_0) = \frac{1}{2} \sum_{n=0}^{\infty} \mathbf{1}_{\inf\{i, a_i \neq b_i\} > n} \tag{4.133}$$

$$= \frac{1}{2}\inf\{i, a_i \neq b_i\}, \tag{4.134}$$

which is what we wanted. □

This proposition allows us to actually compute an unbounded CTW algorithm for $\mathbf{T}$ (specifically with uniform density experts, though): we just have to plug a logarithmic singularity once we reach an interval containing only one data point.

We have not used in practice though.

4.3.4.3 Text compression

The original CTW algorithm on $\{0,1\}$ can be described in the expert tree framework:

- $\mathbf{T}$ is an expert tree from $\{*\}$ to $\{0,1\}$.
- $w_s = 1/2$ for all s.
- Each node s has three children: $0s$ and $1s$, and a special symbol $\S s$ corresponding to the beginning of the word.
- $\mathrm{Tar}(\mathcal{E}_s) = \{0,1\}$, and $\mathrm{Dom}(\mathcal{E}_s) = \{(*,y_1),...,(*,y_n),*|s \text{ is a suffix of } \S^\infty y^n\}$.
- $\mathcal{E}_s$ uses a Krichevsky-Trofimov estimator: $\mathcal{E}_s(i|y^n) = \frac{|\{k|y^{k-1}\in\mathrm{Dom}(\mathcal{E}_s), y_k=i\}|+1/2}{|\{k|y^{k-1}\in\mathrm{Dom}(\mathcal{E}_s)\}|+1}$,

and Definition 4.44 can be used directly for the infinite depth CTW algorithm.

It is also possible to recover the Partition Tree Weighting algorithm at fixed time horizon $n = 2^K$: this time, each node s has two children, $s0$ and $s1$ (notice the suffix notation), and

$$\mathrm{Dom}(\mathcal{E}_s) = \bigcup_{\text{The } K \text{ bits binary expansion of } n \text{ starts with } s} (\{*\}\times\{0,1\})^n \times \{*\}.^{18} \tag{4.135}$$

In other words the left children of $\mathcal{E}_s$ are used for the first half of the points seen by $\mathcal{E}_s$, and the right children are used for the second half.

4.3.4.4 Regression

Let us consider $f : [0,1) \to \mathbb{R}$. Given $x_i \in [0,1)$, we try to predict $f(x_i)$. An expert tree framework for this problem could be for example:

- $\mathbf{T}$ is an expert tree from $[0,1)$ to $\mathbb{R}$
- $w_s = 1/2$ for all s.
- $\mathrm{Dom}(\mathcal{E}_\varepsilon) = [0,1)$, $\mathrm{Tar}(\mathcal{E}_\varepsilon) = Y$.

[18] "K bits binary expansion" means that we complete the binary expansion of n with zeroes on the left.

- Each node s has two children: $s0$ and $s1$.
- If $\text{Dom}(\mathcal{E}_s) = [a, b[$, then $\text{Dom}(\mathcal{E}_{s0} = [a, \frac{a+b}{2}[)$ and $\text{Dom}(\mathcal{E}_{s1} = [\frac{a+b}{2}, b[)$.
- $\text{Tar}(\mathcal{E}_s) = \mathbb{R}$.
- $\mathcal{E}_s$ can be any local regression expert on $\text{Dom}(\mathcal{E}_s)$ (examples of local regression experts will be given in Section 5.1).

The generalization from $f : X \to Y$ is straightforward.

We have not been able to find a general theorem showing that context tree switching behaves better than context tree weighting if we are using a version of CTS that actually switches often. However, as we are now going to show, CTS behaves better than CTW on a reasonable regression problem.

Chapter 5

Comparing CTS and CTW for regression

In this chapter, we are trying to predict the behavior of an unknown function $f : [0, 1] \to \mathbb{R}$. We show that in the noiseless case (i.e., we observe exactly the $f(x_i)$), with local experts fitting a Gaussian with unknown mean and variance to the data, infinite-depth CTS is better than infinite-depth CTW in the sense that its cumulated log-loss is smaller.

5.1 Local experts

In this section, we define the experts that will be used in the context tree to give an estimate of some function f on a *fixed* set $A \subset \mathbb{R}$. These experts use respectively a fixed Gaussian, a Gaussian with fixed variance and variable mean, and a Gaussian with variable mean and variance, which is the expert we will use for the comparison between CTW and CTS.

Information about conjugate priors can be found in [Fin97].

5.1.1 The fixed domain condition

Most of the time, we think of a local regression expert $\mathcal{E}$ from X to Y as attached to a subset D of X: $\mathcal{E}$ will issue a prediction for x_n if and only if $x_n \in D$. In that case, we say that $\mathcal{E}$ is a *fixed domain* expert. More precisely:

Definition 5.1 (Fixed domain)**.** *Let $\mathcal{E}$ be an expert from X to Y with domain D and target T. We say that the domain of $\mathcal{E}$ is* fixed *if there exists $A \subset X$ such that $D = (X \times Y)^* \times A$. In other words, a prediction is issued if and only if the last x seen is in A.*

In that case we will often abuse the definition of the "domain" and write that A is the domain of $\mathcal{E}$.

We will define fixed domain experts below, but the generalization is straightforward.

5.1.2 Blind experts

We first define an expert with fixed mean and fixed precision (inverse of variance).

Definition 5.2. *Let $d \in \mathbb{N}$, $\mu, \tau \in \mathbb{R}$ and $D \subset \mathbb{R}^d$. We call blind expert with domain D, mean μ and precision τ the fixed domain expert $\mathcal{E}$ defined by:* $\mathrm{Dom}(\mathcal{E}) = (\mathbb{R}_d \times \mathbb{R})^* \times D$, $\mathrm{Tar}(\mathcal{E}) = \mathbb{R}$, *and*

$$\mathcal{E}((x,y)|\mathbf{z}) = \frac{\sqrt{\tau}}{\sqrt{2\pi}} e^{-\tau \frac{(y-\mu)^2}{2}} \tag{5.1}$$

for all $x \in D$, $y \in \mathbb{R}$

This expert is not useful by itself, but it will be a building block for the more refined Gaussian expert and Normal-Gamma expert.

5.1.3 Gaussian experts

Gaussian experts try to fit a Gaussian with unknown mean, but fixed precision τ to the data.

Definition 5.3. *Let $d, t, \tau_0 \in \mathbb{N}$, $D \subset \mathbb{R}^d$, and $T \subset \mathbb{R}$, μ_0, $\lambda \in \mathbb{R}$. We call* Gaussian expert *with domain D, regularization μ_0, λ and precision τ_0 (or with variance $\frac{1}{\tau_0}$) the expert noted $\mathcal{E}$ defined by:* $\mathrm{Dom}(\mathcal{E}) = (\mathbb{R}^d \times \mathbb{R})^* D$, $\mathrm{Tar}(\mathcal{E}) = \mathbb{R}$, *and:*

$$\mathcal{E}(\mathbf{z}|\varnothing) = \sqrt{\lambda} \int_\mu \frac{\sqrt{\tau_0}}{\sqrt{2\pi}} e^{-\lambda \frac{\tau_0(\mu-\mu_0)^2}{2}} \mathcal{E}_{\mu,\tau_0}(\mathbf{z}|\varnothing) \mathrm{d}\mu \tag{5.2}$$

for $\mathbf{z} \in (\mathbb{R}^d \times \mathbb{R})^* \times D \times \mathbb{R}$, where $\mathcal{E}_{\mu,\tau_0}$ is the blind expert with domain D, mean μ and precision τ.

In other words, $\mathcal{E}$ is a Bayesian combination of blind experts with weights: $w_\mu = \sqrt{\lambda} \frac{\sqrt{\tau_0}}{\sqrt{2\pi}} e^{-\lambda \frac{\tau_0(\mu-\mu_0)^2}{2}}$.

$$\mathcal{E} = \mathrm{Bayes}_\mu((\mathcal{E}_{\mu,\tau_0}, w_\mu)), \tag{5.3}$$

Remark. The regularization can be interpreted as Jeffreys' prior (which is uniform on μ) with additional λ observations at μ_0.

The predictions are actually simple to compute:

Proposition 5.4 (Computing predictions (conjugate prior))**.** *Let $d, n \in \mathbb{N}$, μ_0, λ, $\tau_0 \in \mathbb{R}$, and let $D \subset \mathbb{R}$. For all $\mathbf{z} \in (D \times \mathbb{R})^n$, if $\mathcal{E}_{\mu_0,\lambda,\tau_0}$ denotes the Gaussian expert with domain $(\mathbb{R} \times \mathbb{R})^* \times D$, regularization μ_0, λ and precision τ_0, we have:*

$$\mathcal{E}_{\mu_0,\lambda,\tau_0}(.|\mathbf{z}) = \mathcal{E}_{\frac{\lambda\mu_0+n\bar{y}}{\lambda+n},\lambda+n,\tau_0}(.|\varnothing), \tag{5.4}$$

where $\bar{x} := \frac{1}{m}\sum_{i=1}^{m} y_i$ is the sample mean, and:

$$\mathcal{E}_{\mu_0,\lambda,\tau_0}(.|x) = \mathcal{N}(\mu_0, \frac{1}{(\lambda+1)\tau_0}). \tag{5.5}$$

for $x \in D$.

5.1.4 Normal-Gamma experts

Normal-Gamma experts try to fit a Gaussian with unknown mean and variance to the data. These are the experts we will use in the expert tree.

Definition 5.5. *Let $d, t \in \mathbb{N}$, $D \subset \mathbb{R}^d$, $\mu_0 \in \mathbb{R}$, λ, α, $\beta > 0$. We call* Normal-Gamma expert *with domain D, with regularization μ_0, λ, α, β the expert $\mathcal{E}$ satisfying:* $\mathrm{Dom}(\mathcal{E}) = (\mathbb{R}^d \times \mathbb{R})^* \times D$, $\mathrm{Tar}(\mathcal{E}) = \mathbb{R}$, *and:*

$$\mathcal{E}(\mathbf{z}|\varnothing) = \int_\mu \int_\tau \frac{\beta^\alpha\sqrt{\lambda}}{\Gamma(\alpha)\sqrt{2\pi}} \tau^{\alpha-\frac{1}{2}} e^{-\beta\tau} e^{-\frac{\lambda\tau(\mu-\mu_0)^2}{2}} \mathcal{E}_{\mu,\tau}(\mathbf{z}|\varnothing)\mathrm{d}\tau\mathrm{d}\mu \tag{5.6}$$

for $\mathbf{z} \in D \times \mathbb{R}$, where $\mathcal{E}_{\mu,\tau}$ is the blind expert with domain D, mean μ and precision τ.

In other words, $\mathcal{E}$ is a Bayesian combination of blind experts with weights $w_{\mu,\tau} = \frac{\beta^\alpha\sqrt{\lambda}}{\Gamma(\alpha)\sqrt{2\pi}} \tau^{\alpha-\frac{1}{2}} e^{-\beta\tau} e^{-\frac{\lambda\tau(\mu-\mu_0)^2}{2}}$:

$$\mathcal{E} = \mathrm{Bayes}_{\mu,\tau}((\mathcal{E}_{\mu,\tau}, w_{\mu,\tau})), \tag{5.7}$$

where $\mathcal{E}_{\mu,\tau}$ is the blind expert with domain D,, mean μ and precision τ.

Consequently, $\mathcal{E}$ is also a Bayesian combination of Gaussian experts with weights $w_\tau = \frac{\beta^\alpha \tau^{\alpha-1} e^{-\beta\tau}}{\Gamma(\alpha)}$ (Gamma distribution on τ):

$$\mathcal{E} = \mathrm{Bayes}_{\tau}((\mathcal{E}_{\mu_0,\lambda,\tau}, w_\tau)), \tag{5.8}$$

where $\mathcal{E}_{\mu_0,\lambda,\tau}$ is the Gaussian expert with domain D, precision τ, and regularizaion μ_0, λ.

Remark. If $\lambda = 2\alpha$, the regularization can be interpreted as Jeffreys' prior with additional α observations at $\mu_0 + \beta/\alpha$ and α observations at $\mu_0 - \beta/\alpha$.

We have the following well-known propositions:

Proposition 5.6 (Conjugate prior)**.** *Let $d, n, m \in \mathbb{N}$, μ_0, λ, α, $\beta \in \mathbb{R}$, $D \subset \mathbb{R}$. Let $\mathcal{E}_{D,\mu_0,\lambda,\alpha,\beta}$ be the Normal-Gamma expert with domain D and target $\mathbb{R}$ with regularization μ_0, λ, α, β. For all $\mathbf{z} \in (D \times \mathbb{R})^n$, we have:*

$$\mathcal{E}_{D,\mu_0,\lambda,\alpha,\beta}(.|\mathbf{z}) = \mathcal{E}_{D,\frac{\lambda\mu_0+n\bar{y}}{\lambda+n},\lambda+n,\alpha+n/2,\beta+\frac{1}{2}\mathrm{Var}(\mathbf{y})+\frac{n\lambda}{n+\lambda}\frac{(\bar{x}-\mu_0)^2}{2}}(.|\varnothing), \tag{5.9}$$

where $\bar{y} := \frac{1}{m}\sum_{i=1}^m y_i$ is the sample mean, and $\mathrm{Var}(\mathbf{y}) := \sum_{i=1}^m (y_i - \bar{y})^2$ is the sample variance.

Proposition 5.7 (Computing predictions)**.** *Let $d, t \in \mathbb{N}$, $D \subset \mathbb{R}^d$, μ_0, λ, α, $\beta \in \mathbb{R}$, let $\mathcal{E}_{D,\mu_0,\lambda,\alpha,\beta}$ be a Normal-Gamma expert on D with regularization μ_0, λ, α, β, and let $z = (x, y) \in D \times \mathbb{R}$. We have:*

$$\mathcal{E}_{D,\mu_0,\lambda,\alpha,\beta}(z|\varnothing) = t_{2\alpha}(y|\mu_0, \frac{\beta(\lambda+1)}{\alpha\lambda}), \tag{5.10}$$

where $t_\alpha(.|\mu, \sigma^2)$ is the Student distribution with α degrees of freedom, with mean μ and variance σ^2 (i.e. $\forall x, t_\alpha(x|\mu, \sigma^2) := t_\alpha(\frac{x-\mu}{\sigma})$).

From now on, we suppose that $\mu_0 = 0$, without loss of generality.

It is easy to check that all the experts defined in this section do not peek out (the blind experts do not peek-out since their prediction is independent of the data, and the other experts are Bayesian combinations of blind experts)[1], so they can be used for expert tree algorithms. Since we want to use infinite depth expert trees, we also have to ensure the experts are compatible. This is the objective of the next section.

5.2 Regularization in expert trees

We consider an infinite depth weighted expert tree $\mathbf{T}$ for regression on the interval $[0, 1]$, as defined in Section 4.3, with a Normal-Gamma expert $\mathcal{E}_s$ at each node s.

We are interested in the relation between the hyperparameters of the the different $\mathcal{E}_s$.

Since we want to use an infinite depth expert tree, the different experts we use have to be compatible. In other words, $\mathcal{E}_s(.|x)$ must not depend on s (if $x \in \mathrm{Dom}(\mathcal{E}_s)$). Since we know (Proposition 5.7) that $\mathcal{E}_{D,\mu_0,\lambda,\alpha,\beta}(y|x) = t_{2\alpha}(y|\mu_0, \frac{\beta(\lambda+1)}{\alpha\lambda})$, we must have the following condition:

Condition 5.8. *We say that $\mathcal{S}$ satisfies condition 5.8 if $\mathcal{S}$ is a set of Normal-Gamma experts, and:*

- $\mu_0(\mathcal{E}) = \mathrm{cst}$ *(we fix it to zero without loss of generality).*

[1] This argument also holds for the non fixed domain versions of the experts.

- $\alpha(\mathcal{E}) = \text{cst}$
- $\frac{\beta(\mathcal{E})(\lambda(\mathcal{E})+1)}{\lambda(\mathcal{E})} = \text{cst} =: B(\mathcal{S})$

for $\mathcal{E} \in \mathcal{S}$.

Proposition 5.9. *Let* $\mathcal{S}$ *be a set of Normal-Gamma experts.* $\mathcal{S}$ *satisfies Condition 5.8 iff for all* $\mathcal{E}, \mathcal{E}' \in \mathcal{S}$, $\mathcal{E}$ *and* $\mathcal{E}'$ *are compatible.*

In particular, an expert tree containing Normal-Gamma experts is a proper tree iff it satisfies Condition 5.8.

Proof. Let $\mathcal{E}, \mathcal{E}' \in \mathcal{S}$, with respective regularizations $(\mu_0, \lambda, \alpha, \beta)$, $(\mu_0', \lambda', \alpha', \beta')$. By Proposition 5.7, $\mathcal{E}(\varnothing) = x \mapsto t_{2\alpha}(y|\mu_0, \frac{\beta(\lambda+1)}{\alpha\lambda})$, and $\mathcal{E}'(\varnothing) = x \mapsto t_{2\alpha'}(y|\mu_0', \frac{\beta'(\lambda'+1)}{\alpha'\lambda'})$.

Since two student distributions are equal iff they have same mean and variance, and same number of degrees of freedom, $\mathcal{E}$ and $\mathcal{E}'$ are compatible iff:

- $\mu_0 = \mu_0'$
- $\alpha = \alpha'$
- $\frac{\beta(\lambda+1)}{\alpha\lambda} = \frac{\beta'(\lambda'+1)}{\alpha'\lambda'}$

This is exactly condition 5.8.

□

Consequently, the most important question when regularizing is: "how should λ (or β) change with $|s|$ (or more precisely, with $\text{Dom}(\mathcal{E}_s)$) ?". This issue will be adressed in Section 5.2.1 below.

We will only consider the cases $\lambda = K|\text{Dom}(\mathcal{E}_s)|^h$, with $K > 0, h \geqslant 0$. In particular, $\lambda = O(1)$ and $\beta = O(1)$ uniformly on the whole tree.

5.2.1 Choosing the regularization

We are interested in the "best" choice of regularization among the $\lambda(\mathcal{E}_s) = K|\text{Dom}(\mathcal{E}_s)|^h$ for the Normal-Gamma experts in an expert tree satisfying Condition 5.8 for regression. To do so, we will study a very simple case: regression for $f : x \mapsto x$ on $[0, 1]$, and sample points x_i such that at time $t = 2^K$, each expert at depth K has seen exactly one point.

We show that in this case, the best choice is $\lambda(\mathcal{E}_s) \propto |\text{Dom}(\mathcal{E}_s)|^2$.

Notation 5.10. *We denote by* $(u_n)_{n\in\mathbb{N}}$ *the sequence defined by:* $u_0 = 0$, *and for all* k, m, *if* $0 \leqslant m < 2^k$, $u_{2^k+m} = \frac{2m+1}{2^{k+1}}$.

If $0 \leqslant m < 2^k$, *we write* $I_{k,m} := [\frac{m}{2^k}, \frac{m+1}{2^k})$.

Notice in particular that the (u_n) are such that u_{2^k+m} is the second point in $I_{k,m}$, and the first point in any $I_{k',m}$ containing u_{2^k+m} with $k' > k$.

We now use these (u_n) to study the behaviour of the CTW algorithm in an expert tree satisfying Condition 5.8.

Let $t = 2^K$, $v_n = (u_n, f(u_n)) = (u_n, u_n)$, and let us consider an expert $\mathcal{E}$ at depth k, with $k \leqslant K$. In particular, $|\mathrm{Dom}(\mathcal{E})| = \frac{1}{2^k}$, and $t_k = \frac{t}{2^k}$ is the number of data points actually seen by $\mathcal{E}$. By Lemma A.9, we have:

$$-\ln \mathcal{E}(v^t|\varnothing) = (\alpha + t_k/2)\ln V_0(t_k,\lambda) + \frac{1+\ln 2\pi}{2}t_k + \ln t_k - \frac{1}{2}\ln\lambda - \alpha\ln\beta + O(1), \tag{5.11}$$

with $V_0(t_k,\lambda) = \frac{2\beta}{t_k+\lambda} + \frac{\lambda}{t_k+\lambda}\frac{\sum_{u_i\in\mathrm{Dom}(\mathcal{E})} u_i^2}{t_k+\lambda} + \left(\frac{t_k}{t_k+\lambda}\right)^2 \mathrm{Var}_{u_i\in\mathrm{Dom}(\mathcal{E})}(u_i)$ (with also $t_k = |\{u_i \in \mathrm{Dom}(\mathcal{E})\}|$).

Now, since $\ln\beta = \ln\lambda + O(1)$, the average cost of a point is:

$$-\frac{1}{t_k}\ln\mathcal{E}(v^t|\varnothing) = (\frac{\alpha}{t_k} + 1/2)\ln V_0(t_k,\lambda) + \frac{1+\ln 2\pi}{2} - \frac{1}{t_k}\ln\frac{1}{t_k} - \frac{1}{t_k}(\frac{1}{2}+\alpha)\ln\lambda + O(\frac{1}{t_k}), \tag{5.12}$$

$$= (\frac{\alpha}{t_k} + 1/2)\ln V_0(t_k,\lambda) - \frac{1}{t_k}(\frac{1}{2}+\alpha)\ln\lambda + O(1) \tag{5.13}$$

$$= \frac{\alpha+1/2}{t_k}(\ln V_0(t_k,\lambda) - \ln\lambda) + \frac{t_k-1}{2t_k}\ln V_0(t_k,\lambda) + O(1) \tag{5.14}$$

$$=: \phi(t_k,\lambda) + O(1). \tag{5.15}$$

Lemma 5.11. *We have, if $\lambda = O(1)$: $t_k > 1$:*

$$\ln V_0(t_k,\lambda) = \max\left[\ln\lambda - \ln t_k, -2k\ln 2\right] + O(1) \qquad \text{if } t_k > 1, \tag{5.16}$$

$$\ln V_0(t_k,\lambda) = \ln\lambda + O(1) \qquad \text{if } t_k = 1 \tag{5.17}$$

where $V_0(t_k,\lambda) = \frac{2\beta}{t_k+\lambda} + \frac{\lambda}{t_k+\lambda}\frac{\sum_{u_i\in\mathrm{Dom}(\mathcal{E})} u_i^2}{t_k+\lambda} + \left(\frac{t_k}{t_k+\lambda}\right)^2 \mathrm{Var}_{u_i\in\mathrm{Dom}(\mathcal{E})}(u_i)$.

Proof. We have, for any $a,b,c > 0$, $\ln(\max(a,b,c)) \leqslant \ln(a+b+c) \leqslant \ln(3\max(a,b,c)) = \ln(\max(a,b,c)) + \ln 3$. In particular $\ln(a+b+c) = \ln(\max(a,b,c)) + O(1) = \max(\ln(a,b,c)) + O(1)$.

We now apply this to the three terms in $V_0(t+\lambda)$:

$$\ln\frac{2\beta}{t_k+\lambda} = \ln\frac{B\lambda}{(\lambda+1)t_k(1+\frac{\lambda}{t_k})} = \ln(\frac{\lambda}{t_k}) + \ln\left(\frac{B}{(\lambda+1)(1+\lambda/t_k)}\right) = \ln\lambda - \ln t_k + O(1). \tag{5.18}$$

Since for all i, $u_i \in [0,1]$, we have $\sum_{u_i\in\mathrm{Dom}(\mathcal{E})} u_i^2 \leqslant t$, so:

$$\ln\left(\frac{\lambda}{t_k+\lambda}\frac{\sum_{u_i\in\mathrm{Dom}(\mathcal{E})} u_i^2}{t_k+\lambda}\right) \leqslant \ln\left(\frac{\lambda t_k}{t_k^2(1+\frac{\lambda}{t_k})^2}\right) = \ln\lambda - \ln t_k + O(1). \tag{5.19}$$

If $t_k = 1$, $\mathrm{Var}_{u_i \in \mathrm{Dom}(\mathcal{E}_i)}(u_i) = 0$ and the lemma follows.

If $t_k > 1$, suppose without loss of generality that $\mathrm{Dom}(\mathcal{E}) = [0, \frac{1}{2^k})$, so that $\{u_i \in \mathrm{Dom}(\mathcal{E}_i)\} = \{\frac{m}{2^K} | m \in [\![0, 2^{K-k} - 1]\!]\}$. Let us consider the $(2^K u_i)$, and let us write $A = 2^{K-k-1}$. Their average is $\bar{u} = \frac{A}{2}$, so

$$\mathrm{Var}_{u_i \in \mathrm{Dom}(\mathcal{E}_i)}(2^K u_i) = \frac{1}{A+1} \sum_{m=0}^{A} (m - A/2)^2 \tag{5.20}$$

$$= \frac{1}{A+1} \sum_{m=0}^{A} m^2 - \frac{A}{2} \sum_{m=0}^{A} m + (A+1)A^2/4 \tag{5.21}$$

$$= \frac{A(2A+1)}{6} - \frac{A^2}{2} + \frac{A^2}{4} \tag{5.22}$$

$$= \frac{A^2}{12} + \frac{A}{6}. \tag{5.23}$$

So $\mathrm{Var}_{u_i \in \mathrm{Dom}(\mathcal{E}_i)}(u_i) = \frac{1}{2^{2K}} \mathrm{Var}_{u_i \in \mathrm{Dom}(\mathcal{E}_i)}(2^K u_i) = \frac{2^{-2k-2}}{12} + \frac{2^{-k-K-1}}{6}$. Since $k \leqslant K$, we finally have: $\ln \mathrm{Var}_{u_i \in \mathrm{Dom}(\mathcal{E}_i)}(u_i) = -2k \ln 2 + O(1)$, and

$$\ln \left(\frac{t_k}{t_k + \lambda} \right)^2 \mathrm{Var}_{u_i \in \mathrm{Dom}(\mathcal{E})}(u_i) = \ln(\frac{1}{(1 + \frac{\lambda}{t_k})^2}) + \ln \mathrm{Var}_{u_i \in \mathrm{Dom}(\mathcal{E}_i)}(u_i) \tag{5.24}$$

$$= -2k \ln 2 + O(1). \tag{5.25}$$

The lemma follows □

Plugging $t_k = 1$ in (5.14) immediately yields a $O(1)$ average loss, for any regularization. As we will see, this worse than what we can obtain with $t_k > 1$:

- Suppose $\lambda(\varepsilon) = \varepsilon^h$, with $h > 2$. Then $V_0(t_k, \lambda) = -2k \ln 2 + O(1)$. In particular, $h \mapsto \phi(t_k, \frac{1}{2^{kh}})$ is decreasing on $(2, +\infty)$. Intuitively, increasing h over 2 increases the cost for each expert ($-\ln \lambda$ term), but does not improve their performance ($-\ln V_0$ term).
- Suppose $\lambda(\varepsilon) = \varepsilon^h$, with $h < 2$. In that case, $\ln \lambda - \ln t_k = (-kh - K + k) \ln 2 = -(K + k(h-1)) \ln 2$. Let us study how the average loss of $\mathcal{E}$ varies with its depth k.

  — For $k(3-h) \geqslant K$, $\ln \lambda - \ln t_k \geqslant -2k \ln 2$, so $\ln V_0(t_k, \lambda) = \ln \lambda - \ln t_k$. In particular, (5.14) yields:

$$-\frac{1}{k} \ln \mathcal{E}(v^t | \varnothing) = \frac{t_k - 1}{2t_k} (\ln \lambda - \ln t_k) \tag{5.26}$$

$$= -\frac{t_k - 1}{2t_k} (K + k(h-1)) \ln 2 + O(1). \tag{5.27}$$

The dominant term is therefore $-\frac{1}{2}(K + k(h-1))\ln 2$ (if it is not also in the $O(1)$). In particular:

- It is decreasing in k if $2 > h \geqslant 1$, so the optimal choice is $k = \frac{K}{3-h}$, and the loss is (up to terms of lesser order) $-\frac{1}{2}(K + K\frac{h-1}{3-h})\ln 2 = -\frac{h-1}{3-h}\ln t$, with $\frac{h-1}{3-h} < \frac{2-1}{3-2} = 1$.
- It is increasing in k if $h \leqslant 1$, so the optimal choice is $k = 0$, and the loss is (up to terms of lesser order) $-\frac{1}{2}K\ln 2 = -\frac{1}{2}\ln t$.

—For $k(3-h) \leqslant K$, $\ln\lambda - \ln t_k \leqslant -2k\ln 2$, so $\ln V_0(t_k, \lambda) = -2k\ln 2$. In this case, we show that the best loss is achieved for k close to K (i.e. $k = \frac{K}{3-h}$), and then, as seen before, the loss is $-\frac{h-1}{3-h}\ln t$ if $h \geqslant 1$, and $-\frac{1}{2}\ln t$ otherwise.

Indeed, (5.14) yields

$$\phi(t_k, \lambda) = \frac{\alpha + 1/2}{t_k}(-2k\ln 2 + kh\ln 2) - \frac{t_k - 1}{2t_k}2k\ln 2 \tag{5.28}$$

$$= \left[-k + k\frac{2^k}{t} - (\alpha + 1/2)(2-h)k\frac{2^k}{t}\right]\ln 2) \tag{5.29}$$

$$= \left[-k + k\frac{2^k}{t}(1 - (\alpha + 1/2)(2-h))\right]\ln 2, \tag{5.30}$$

so

$$\frac{1}{\ln 2}\frac{\mathrm{d}\phi(t_k, \lambda)}{\mathrm{d}k} = -1 + \frac{2^k}{t}(1 - (\alpha + 1/2)(2-h)) + k\frac{2^k}{t}(1 - (\alpha + 1/2)(2-h))\ln 2 \tag{5.31}$$

$$= -1 + \frac{2^k}{t}(\mathrm{cst} * k\ln 2 + \mathrm{cst}) \tag{5.32}$$

But since $k \leqslant \frac{K}{3-h} \leqslant K$, we have $2^k \leqslant t^{\frac{1}{3-h}} =: t^{1-\nu}$, with $\nu = 1 - \frac{1}{3-h} > 0$; and $k\ln 2 \leqslant K\ln 2 = \ln t$. So

$$\frac{1}{\ln 2}\frac{\mathrm{d}\phi(t_k, \lambda)}{\mathrm{d}k} \leqslant -1 + \mathrm{cst}\frac{\ln t}{t^\nu}. \tag{5.33}$$

Consequently, for t large enough $\frac{\mathrm{d}\phi(t_k,\lambda)}{\mathrm{d}k}$ is negative on $[0, \frac{K}{3-h})$, so the optimal k is $\frac{K}{3-h}$, with an average loss of $-\frac{h-1}{3-h}\ln t$.

So finally, the best regularization for a linear function with a balanced sequence of sample points is $\lambda(\varepsilon) \propto |\mathrm{Dom}(\mathcal{E})|^2$. This is the regularization we will use from now on.

At first glance, it would have seemed reasonable to choose a regularization proportional to $\lambda(\varepsilon) \propto \varepsilon$, because it is the only regularization that could be described as virtual points seen by the whole tree.

However, since we suppose we are regressing Lipschitz functions, we know that the variation of f in an interval of size ε is of order ε, so a better regularization would ensure that the expected deviation of $\mathcal{E}_s$ after having seen on point is of order ε. Since the variance is roughly proportional to $\lambda(\varepsilon)$, we must have $\lambda(\varepsilon) = \varepsilon^2$. We expect that in general, if we are regressing a h-Hölder function, the best regularization is $\lambda(\varepsilon) = \varepsilon^{2h}$.

5.3 Regret bounds in the noiseless case

We are now going to prove that for an infinite binary weighted expert tree $\mathbf{T}$ satisfying Condition 5.8, if the sequence of sample points (x_i) is sufficiently well-behaved:

- The loss of the context tree weighting algorithm corresponding to $\mathbf{T}$ (and more generally, of *any* Bayesian combination of Normal–Gamma experts) is greater than $-t \ln t + \frac{1}{2} t \ln \ln t + O(t)$
- The loss of the context tree switching algorithm corresponding to $\mathbf{T}$ is lower than $-t \ln t + O(t)$.[2] Using the edgewise version of the algorithm broadens the class of functions for which the bound holds.

The idea behind the proofs is always the following: to generate the image $f(x_i)$ of a new point x_i, CTS can use the deepest expert that has seen x_i and at least one other data point (provided the expert is not "too deep": the cost of specifying the tree must not be too high), whereas CTW cannot (from a generative point of view, CTW consists in choosing a given tree, and using it to generate the data).

5.3.1 CTS

The expert trees we use to achieve the bound for CTS and ECTS are "well-behaved" infinite binary trees. More precisely:

Definition 5.12. *Let $I \subset \mathbb{R}$ be an interval, and let $\mathbf{T}$ be an expert tree with $\mathrm{Dom}(\mathcal{E}_\varepsilon) = I$.*

We say that $\mathbf{T}$ is n-balanced *if:*

- *Each node of $\mathbf{T}$ has exactly n children.*
- *For all $s \in \mathbf{T}$, the domain of $\mathcal{E}_s$ has is fixed.*
- *For all $s \in \mathbf{T}$, $\mathrm{Dom}(\mathcal{E}_s)$ is an interval.*
- *For all $s \in \mathbf{T}$ and for all $t, t \in c(s)$, $|\mathrm{Dom}(\mathcal{E}_t)| = |\mathrm{Dom}(\mathcal{E}_{t'})|$*

[2] Under a condition on the switching that is satisfied by the Switch prior from Definition 4.16.

In particular, a n-balanced tree is infinite.

The bound we obtained for non edgewise CTS holds only in a simple case: we choose a sequence of sample points such that at time 2^K, each expert at depth K has seen exactly one data point.

Theorem 5.13 (Lower bound for CTS regret)**.** *Let (u_n) be as in Notation 5.10.*

Let $K_f \in \mathbb{R}^+$, let f be a K_f-Lipschitz function on $[0,1]$, and let $t \in \mathbb{N}$.

Let $\mathbf{T}$ be a proper 2-balanced weighted expert tree such that $\mathrm{Dom}\mathcal{E}_\varepsilon = [0,1]$ and $h := \inf\{\mu_t(J) | t \in \mathbf{T}, (J \sim_3 1^\infty \text{ or } J = 0^\infty)\} > 0$.

If for any expert $\mathcal{E}_s$, for any $(z_1, z_2) \in \mathrm{Tar}(\mathcal{E}_s)^{\mathbb{N}}$, we have:

$$-\ln \mathcal{E}_s(z_1|\varnothing) \leqslant C(f) \tag{5.34}$$

and,

$$-\ln \mathcal{E}_s(z_2|z_1) \leqslant C(f) - |s| \ln 2 =: L_{|s|}, \tag{5.35}$$

then,

$$-\ln \mathrm{CTS}_{\mathbf{T}}(v^t|\varnothing) \leqslant -t \ln t + O(t), \tag{5.36}$$

where for all n, $v_n := (u_n, f(u_n))$

Proof. Let K such that $t \in (2^{K-1}, 2^K]$.

Let $\mathcal{I}$ be a switching pattern on $\mathbf{T}$ such that $\mathcal{I}_s = (001^\infty)$ for $|s| \leqslant K-2$ and $\mathcal{I}_s = (0^\infty)$ for $|s| = K-1$.

By Lemma 4.45, we have:

$$-\ln \mathrm{CTS}_{\mathbf{T}}(v^t|\varnothing) \leqslant -\ln \mathcal{E}_{\mathcal{I}}(v^t|\varnothing) - (2^K - 1) \ln h = -\ln \mathcal{E}_{\mathcal{I}}(v^t|\varnothing) + O(t). \tag{5.37}$$

By definition of $\mathcal{I}_\varepsilon$, we have

$$\mathcal{E}_{\mathcal{I}}(v_1|v_0) = \mathcal{E}_\varepsilon(v_1|v_0). \tag{5.38}$$

Now, by Lemma A.6, for all $k > 0$, for all $0 \leqslant m < 2^k$, u_{2^k+m} is at least the third point in all of the $I_{k',m'}$ such that $k' < k$ and $u_{2^k+m} \in I_{k',m'}$, and it is the second point in $I_{k,m}$, so we have $\mathcal{E}_{\mathcal{I}}(v_{2^k+m}|v^{2^k+m-1}) = \mathcal{E}_{s(k,m)}(v_{2^k+m}|v^{2^k+m-1})$, where $s(k,m)$ is the node of $\mathbf{T}$ satisfying $\mathrm{Dom}(\mathcal{E}_s) = I_{k,m}$. Finally, if t_0 is the index of the first point in $I_{k,m}$, $\mathcal{E}_{s(k,m)}(v_{2^k+m}|v^{2^k+m-1}) = \mathcal{E}_{s(k,m)}(v_{2^k+m}|v_{t_0})$, since the experts do not peek out.

Since by construction, $|s(k,m)| = k$, we have $\mathcal{E}_{\mathcal{I}}(v_{2^k+m}|v^{2^k+m-1}) = \mathcal{E}_{s(k,m)}(v_{2^k+m}|v_{t_0}) \leqslant L_k$.

Consequently, we have (up to the term corresponding to v_0, which is a constant):

$$-\ln \mathrm{CTS}_{\mathbf{T}}(v^t|\varnothing) \leqslant -\ln \mathcal{E}_{\mathcal{I}}(v^t|\varnothing) + O(t) = \sum_{k=0}^{K-2} 2^k L_k + (t - 2^{K-1}) L_{K-1} + O(t) \tag{5.39}$$

$$\leqslant -\ln 2 \sum_{k=0}^{K-2} 2^k k - (t - 2^{K-1})(K-1)\ln 2 + O(t) \tag{5.40}$$

and by applying Corollary A.2, we find:

$$-\ln \mathrm{CTS}_{\mathbf{T}}(v^t|\varnothing) \leqslant -t \ln t + O(t), \tag{5.41}$$

which is what we wanted. □

Corollary 5.14. *Let $K_f \in \mathbb{R}^+$, let f be a K_f-Lipschitz function on $[0,1]$, let (u_n) be as in Notation 5.10.*

Let $\mathbf{T}$ be a 2-balanced weighted expert tree with $\mathrm{Dom}\mathcal{E}_\varepsilon = [0,1]$, satisfying Condition 5.8 with $\lambda(\mathcal{E}_s) = |\mathrm{Dom}(\mathcal{E}_s)|^2$, and such that there exists $h > 0$ such that for any node s of $\mathbf{T}$, $\min(\mu_s(01^\infty), \mu_s(0^\infty)) \geqslant h$.

We have:

$$-\ln \mathrm{CTS}_{\mathbf{T}}(v^t|\varnothing) \leqslant -t \ln t + O(t). \tag{5.42}$$

where for all n, $v_n := (u_n, f(u_n))$

Proof. By Proposition 5.9, a weighted expert tree satisfying Condition 5.8 is a proper tree, and by Lemma A.10, the Normal-Gamma experts satisfy conditions of Theorem 5.13. □

As discussed at the beginning of 4.3.3, the problem with CTS (or at least with the proof of Theorem 5.13) is that the data points have to alternate exactly between the left and right, for *all* experts.

Edgewise CTS offers the same guaranteed performance on a larger class of functions.

5.3.2 ECTS

For ECTS to work properly, we need sequences with "low discrepancy". More precisely,

Definition 5.15. *Let I_0 be a bounded interval of $\mathbb{R}$. Let $(x_n)_{n\in\mathbb{N}} \in I_0^{\mathbb{N}}$. Let $K > 1$.*

For any interval $I \subset I_0$, for any $t \in \mathbb{R}$, let $N_t(I) := |\{k | 1 \leqslant k \leqslant t, x_k \in I\}|$ be the number of points of (x_n) having falled in I before time t.

- *We say that* (x_n) *is* well K-balanced *if for any interval* $I \subset I_0$*, for any* $t \in \mathbb{N}$

$$\left\lfloor \frac{1}{K}|I|t \right\rfloor \leqslant N_t(I) \leqslant \max(1, K|I|t). \tag{5.43}$$

- *We say that* (x_n) *is* upper well K-balanced *if for any interval* $I \subset I_0$*, for any* $t \in \mathbb{N}$

$$N_t(I) \leqslant \max(1, K|I|t). \tag{5.44}$$

We say that (x_n) *is well balanced (resp. upper well balanced) if there exists* K *such that* (x_n) *is well* K*-balanced (resp. upper well* K*-balanced).*

Remark 5.16. *Let* $(u_n) \in [0,1]^{\mathbb{N}}$ *defined by* $u_0 = 0$*, and for all* k, m*, if* $0 \leqslant m < 2^k$*,* $u_{2^k+m} = \frac{2m+1}{2^{k+1}}$ *(Notation 5.10).* (u_n) *is well 4-balanced.*

Proof. For $t \in (2^{K-1}, 2^K]$, there are less points in a given I than for $t = 2^K$, and for $t = 2^K$, the distance between a point and its closest neighbor is $\frac{1}{2^K}$, so if $|I| < \frac{1}{2^K}$, I contains at most one point, and if $|I| \geqslant \frac{1}{2^K}$, I contains at most $2^K|I| + 1 \leqslant 2 * 2^K|I| \leqslant 2 * 2t|I|$, so (u_n) is upper well 4-balanced. Conversely, for $t \in [2^K, 2^{K+1})$, there are more points in a given I than for $t = 2^K$, and there are $\lfloor 2^K|I| \rfloor$ elements of u_n in the adherence $\bar{I}$ of I, so there are at least $\lfloor 2^K|I| \rfloor / 2 \geqslant \lfloor 2^K|I|/4 \rfloor$ in I, so (u_n) is well 4-balanced. □

Theorem 5.17 (Lower bound for ECTS)**.** *Let* $K_f \in \mathbb{R}^+$*, let* f *be a* K_f*-lipschitz function on* $[0,1]$*.*

Let $\mathbf{T}$ *be an proper infinite 2-balanced edgewise weighted expert tree such that* $\mathrm{Dom}\mathcal{E}_\varepsilon = [0,1]$ *and* $h := \inf\{\mu_{s\to t}(J) | s \in \mathbf{T}, t \in c(s), (J \sim_2 1^\infty \text{ or } J = 0^\infty)\} > 0$ *.*

If for all $s \in \mathbf{T}$*, for any* $(x_i) \in \mathrm{Dom}(\mathcal{E}_s)^{\mathbb{N}}$*, the local experts satisfy:*

$$-\frac{1}{n} \ln \mathcal{E}_s(z_1, ..., z_n | z'_1, ..., z'_m) \leqslant C(f,m) - |s| \ln 2 =: L_{|s|,m} \tag{5.45}$$

for all $n \in \mathbb{N}$*,* $m \in \mathbb{N}^*$*. Then, for any well balanced sequence* $(x_i) \in (\mathrm{Dom}\mathcal{E}_\varepsilon)^{\mathbb{N}}$ *we have:*

$$-\ln \mathrm{ECTS}_{\mathbf{T}}(z^t | \varnothing) \leqslant -t \ln t + O(t), \tag{5.46}$$

where $z_n := (x_n, f(x_n))$*.*

Proof. Let $K \in \mathbb{N}$ such that $t \in (2^{K-1}, 2^K]$, let $(x_n) \in [0,1]^{\mathbb{N}}$. Let λ such that (x_n) is well λ-balanced, and let $\mathcal{I} = \mathcal{I}(\mathbf{x})$ be an edgewise switching pattern such that for any $s \in \mathbf{T}$, and $t \in c(s)$

- $\mathcal{I}_{s\to t} = 0^\infty$ if $|s| = K - 1$.
- $\mathcal{I}_{s\to t} = 01^\infty$ otherwise.

In other words, to predict x_n, $\mathcal{E}_{\mathcal{I}}$ tries to use the deepest expert (at maximum depth $K-1$) that has seen at least one other data point before x_n (Lemma A.3 in the Appendix).

By Lemma 4.46 applied to the root of $\mathbf{T}$, we have:

$$-\ln \mathrm{ECTS}_{\mathbf{T}}(z^t|\varnothing) \leqslant -\ln \mathcal{E}_{\mathcal{I}}(z^t|\varnothing) - (2^{K+1}-2)\ln h = -\ln \mathcal{E}_{\mathcal{I}}(z^t|\varnothing) + O(t). \tag{5.47}$$

If we denote by s_n the node corresponding to the expert used at time n, we have:

$$-\ln \mathcal{E}_{\mathcal{I}}(z^t|\varnothing) = -\sum_{n=1}^{t} \ln \mathcal{E}_{s_n}(x_n|x^{n-1}) = -\sum_{\substack{n=1 \\ |s_n|<K-1}}^{t} \ln \mathcal{E}_{s_n}(x_n|x^{n-1}) - \sum_{\substack{n=1 \\ |s_n|=K-1}}^{t} \ln \mathcal{E}_{s_n}(x_n|x^{n-1}). \tag{5.48}$$

By Lemma A.3, s_n satisfies $x_n \in \mathrm{Dom}(\mathcal{E}_{s_n})$, $x^{n-1} \cap \mathrm{Dom}(\mathcal{E}_{s_n}) \neq \varnothing$ and either $|s_n| = K-1$ or $x^{n-1} \cap \mathrm{Dom}(\mathcal{E}_{t_n}) = \varnothing$, where t_n is the child of s_n containing x_n.

In particular, for any $s \in \mathbf{T}$: if $|s| = K-1$ and $n > n(s)$ where $n(s) := \inf\{i \in \mathbb{N} || x_i \in \mathrm{Dom}(\mathcal{E}_s)|\}$ is the index of the first point of $\mathbf{x}$ belonging to $\mathrm{Dom}(\mathcal{E}_s)$, then $x_n \in \mathrm{Dom}(\mathcal{E}_{s_n}) \Leftrightarrow s_n = s$.

If we regroup the terms corresponding to the same experts in the second sum, we see that:

$$-\sum_{\substack{n=1 \\ |s_n|=K-1}}^{t} \ln \mathcal{E}_{s_n}(x_n|x^{n-1}) = -\sum_{\substack{s\in\mathbf{T} \\ |s|=K-1}} \ln \mathcal{E}_s(x^{t(s)+1:t}|x^{t(s)}) \tag{5.49}$$

because if $|s| = K-1$, every point falling in $\mathrm{Dom}(\mathcal{E}_s)$ after $t(s)$ is predicted with $\mathcal{E}_s$,

$$\tag{5.50}$$

$$= -\sum_{\substack{s\in\mathbf{T} \\ |s|=K-1}} \ln \mathcal{E}_s(x^{t(s)+1:t} \cap \mathrm{Dom}(\mathcal{E}_s)|x^{t(s)} \cap \mathrm{Dom}(\mathcal{E}_s)) \tag{5.51}$$

because the local experts do not peek out (By hypothesis: $\mathbf{T}$ is a proper tree),

$$\tag{5.52}$$

$$= -\sum_{\substack{s\in\mathbf{T} \\ |s|=K-1}} \ln \mathcal{E}_s(x^{t(s)+1:t} \cap \mathrm{Dom}(\mathcal{E}_s)|x_{t(s)}), \tag{5.53}$$

by definition of $t(s)$.

So by hypothesis (5.45), we have:

$$-\sum_{\substack{n=1\\|s_n|=K}}^{t} \ln \mathcal{E}_{s_n}(x_n|x^{n-1}) = -\sum_{\substack{s\in\mathbf{T}\\|s|=K}} \ln \mathcal{E}_s(x^{t(s)+1:t}\cap \mathrm{Dom}(\mathcal{E}_s)|x_{t(s)}) \leqslant \sum_{\substack{s\in\mathbf{T}\\|s|=K-1}} n_s L_{K-1,1} = n_{K-1}L_{K-1,1} \tag{5.54}$$

where $n_s = |x^{t(s)+1:t}\cap \mathrm{Dom}(\mathcal{E}_s)|$ is the number of times $\mathcal{E}_s$ has been used, and $n_{K-1} = \sum_{\substack{s\in\mathbf{T}\\|s|=K-1}} n_s$ is the number of times an expert at depth $K-1$ has been used.

Now, let us consider the term

$$-\sum_{\substack{n=1\\|s_n|<K-1}}^{t} \ln \mathcal{E}_{s_n}(x_n|x^{n-1}) = -\sum_{\substack{n=1\\|s_n|<K-1}}^{t} \ln \mathcal{E}_{s_n}(x_n|x^{n-1}\cap \mathrm{Dom}(\mathcal{E}_{s_n})). \tag{5.55}$$

The s_n in this sum satisfy: $x_n \in \mathrm{Dom}(\mathcal{E}_{s_n})$, $x^{n-1}\cap \mathrm{Dom}(\mathcal{E}_{s_n}) \neq \varnothing$, and $x^{n-1}\cap \mathrm{Dom}(\mathcal{E}_{t_n}) = \varnothing$, where t_n is the child of s_n containing x_n.

In particular, since $x^{n-1}\cap \mathrm{Dom}(\mathcal{E}_{t_n}) = \varnothing$, if we denote by $t'_n \in \mathbf{T}$ the child of s_n that is not t_n, we have: $x^{n-1}\cap \mathrm{Dom}(\mathcal{E}_{s_n}) = x^{n-1}\cap \mathrm{Dom}(\mathcal{E}_{t'_n})$.

Now, by Proposition A.4, $x^{n-1}\cap \mathrm{Dom}(\mathcal{E}_{t_n}) = \varnothing$ implies $1 \leqslant x^{n-1}\cap \mathrm{Dom}(\mathcal{E}_{t'_n}) \leqslant \lambda^2$, so by hypothesis (5.45), if we write $L_n := \max_{1\leqslant m\leqslant\lambda^2} L_{n,m}$:[3]

$$-\sum_{\substack{n=1\\|s_n|<K-1}}^{t} \ln \mathcal{E}_{s_n}(x_n|x^{n-1}\cap \mathrm{Dom}(\mathcal{E}_{s_n})) \leqslant \sum_{\substack{n=1\\|s_n|<K-1}}^{t} L_{|s_n|} = \sum_{k=0}^{K-2} n_k L_k, \tag{5.56}$$

where $n_k = |\{s_n, |s_n| = k\}|$ is the number of times an expert at depth k has been used.

So finally, if we regroup the two sums:

$$-\ln \mathcal{E}_{\mathcal{I}}(z^t|\varnothing) = -\sum_{\substack{n=1\\|s_n|<K-1}}^{t} \ln \mathcal{E}_{s_n}(x_n|x^{n-1}) - \sum_{\substack{n=1\\|s_n|=K-1}}^{t} \ln \mathcal{E}_{s_n}(x_n|x^{n-1}) \leqslant \sum_{k=0}^{K-1} n_k L_k \tag{5.57}$$

But any given node $s\in\mathbf{T}$ satisfying $|s| < K-1$ can appear only once in the sum (5.56) (after a node appears in the sum, its two children contain at least one data point, and the condition $x^{n-1}\cap \mathrm{Dom}(\mathcal{E}_{t_n}) = \varnothing$ cannot be satisfied anymore), so we must have $n_k \leqslant 2^k$ for all $k \leqslant K-2$.

Since $k \mapsto L_k$ is decreasing and $\sum n_k = t$ is fixed, $\sum_{k=0}^{K-1} n_k L_k$ is clearly maximal for $n_k = 2^k$ for all $k < K-1$: If one of the n_k, say k_0, is smaller than

[3] So by definition, $L_{K-1,1}$ from (5.45) is smaller than L_{K-1}

2^k, we must have $n_{K-1} > 2^{K-1}$. Therefore, $n_{K-1} \leftarrow n_{K-1}-1; n_{k_0} \leftarrow n_{k_0}+1$; yields a larger sum. Consequently, up to the first point, we have

$$-\ln \mathrm{ECTS}_{\mathbf{T}}(z^t|\varnothing) \leqslant -\ln \mathcal{E}_{\mathcal{I}}(z^t|\varnothing) + O(t) \leqslant \sum_{k=0}^{K-2} 2^k L_k + (t-2^{K-1})L_{K-1} + O(t) \tag{5.58}$$

$$\leqslant -\ln 2 \sum_{k=0}^{K-2} k2^k - (t-2^{K-1})(K-1)\ln 2 + O(t) \tag{5.59}$$

$$\leqslant -t\ln t + O(t), \tag{5.60}$$

by Corollary A.2, which is what we wanted.

□

Corollary 5.18. *Let $K_f \in \mathbb{R}^+$. Let f be a K_f-Lipschitz function on $[0,1]$. Let $K \in \mathbb{N}$, and $t = 2^K$. Let $\mathbf{T}$ be an infinite 2-balanced edge-wise weighted expert tree with $\mathrm{Dom}\mathcal{E}_\varepsilon = [0,1]$, satisfying Condition 5.8 with $\lambda(\mathcal{E}_s) = |\mathrm{Dom}(\mathcal{E}_s)|^2$, and such that $h := \inf\{\mu_{s\to t}(J) | s \in \mathbf{T}, t \in c(s), (J \sim_2 1^\infty \text{ or } J = 0^\infty)\} > 0$.[4] Then, we have:*

$$-\ln \mathrm{ECTS}_{\mathbf{T}}(z^t|\varnothing) \leqslant -t\ln t + O(t), \tag{5.61}$$

for any well balanced sequence $(x_i) \in (\mathrm{Dom}\mathcal{E}_\varepsilon)^{\mathbb{N}}$, where $z_n := (x_n, f(x_n))$.

Proof. By Proposition 5.9 Condition 5.8 ensures that $\mathbf{T}$ is a proper tree, and by Lemma A.11, the Normal-Gamma experts satisfy the condition of Theorem 5.17. □

5.3.3 CTW

It is possible to obtain a lower bound on the loss of *any* Context Tree Weighting algorithm, also holding on Bayesian combination of histograms, provided the observed is only *upper* well balanced (we only need to prevent deep experts from seeing "too many" data points).

The bound for the Context Tree Weighting algorithm, and the Bayesian histograms follows from the theorem below:

Theorem 5.19 (Average loss)**.** *Let $K_f \in \mathbb{R}^+$, let f be a K_f-Lipschitz function on $[0,1]$.*

Let $\mathcal{S}$ be a set of Normal-Gamma experts satisfying condition 5.8, with $\lambda(\mathcal{E}) = |\mathrm{Dom}(\mathcal{E})|^2$ and $\mathcal{E} \mapsto |\mathrm{Dom}(\mathcal{E})|$ bounded on $\mathcal{S}$.

We have, for any $\mathcal{E} \in \mathcal{S}$, for any upper well balanced sequence (x_n):

[4] We recall that this condition is achieved by the switch distribution from [vEGdR12].

$$-\frac{1}{t_{\mathcal{E}}}\ln \mathcal{E}(z^t|\varnothing) \geqslant -\ln t + \frac{1}{2}\ln\ln t + O(1), \tag{5.62}$$

uniformly on $\mathcal{S}$, where $z_t = (x_t, f(x_t))$, and $t_{\mathcal{E}} := |z^t \cap \mathrm{Dom}(\mathcal{E})|$ is the number of points predicted by the expert $\mathcal{E}$.

Proof. We write $\varepsilon := |\mathrm{Dom}(E)|$. Since Normal-Gamma experts do not peek out, we have by Lemma A.12:

$$\frac{1}{t_{\mathcal{E}}}\mathcal{E}(z^t|\varnothing) = \frac{1}{t_{\mathcal{E}}}\mathcal{E}(z^t \cap \mathrm{Dom}(\mathcal{E})|\varnothing) \geqslant (1-\frac{1}{t_{\mathcal{E}}})\ln\varepsilon - \frac{1}{2}\ln t_{\mathcal{E}} + O(1) \quad =: \phi_\varepsilon(t_{\mathcal{E}}) + O(1) \tag{5.63}$$

We have $\frac{\mathrm{d}\phi_\varepsilon}{\mathrm{d}t_{\mathcal{E}}} = \frac{\ln\varepsilon}{t_{\mathcal{E}}^2} - \frac{1}{2t_{\mathcal{E}}} = \frac{2\ln\varepsilon - t_{\mathcal{E}}}{2t_{\mathcal{E}}^2}$: ϕ_ε is increasing on $[1, 2\ln\varepsilon]$ (if non empty) and decreasing on $[2\ln\varepsilon, \infty)$, so its minimum on any segment $[1, a]$ is attained either at 1 or at a.

Since (x_n) is upper well balanced, there exists K such that $t_{\mathcal{E}} \leqslant \max(1, K\varepsilon t)$. So we have:

$$\phi_\varepsilon(t_{\mathcal{E}}) \geqslant \min(\phi_\varepsilon(1), \phi_\varepsilon(\max(1, K\varepsilon t))) = \min(0, \phi_\varepsilon(K_1\varepsilon t)) \tag{5.64}$$

We are therefore interested in the minimum value of $\varepsilon \mapsto \phi_\varepsilon(K\varepsilon t)$. We have

$$\phi_\varepsilon(K\varepsilon t) = (1-\frac{1}{K\varepsilon t})\ln\varepsilon - \frac{1}{2}\ln(K\varepsilon t) \tag{5.65}$$

$$= \frac{1}{2}\ln\varepsilon - \frac{1}{2}\ln t - \frac{\ln\varepsilon}{K\varepsilon t} + O(1). \tag{5.66}$$

If we set $u = \frac{\varepsilon t}{\ln t}$, we find:

$$\phi_\varepsilon(K\varepsilon t) = \frac{1}{2}\ln u - \frac{1}{2}\ln t + \frac{1}{2}\ln\ln t - \frac{1}{2}\ln t - \frac{\ln(u\ln t) - \ln t}{Ku\ln t} + O(1) \tag{5.67}$$

$$= -\ln t + \frac{1}{2}\ln\ln t + \frac{1}{2}\ln u + \frac{1}{Ku} - \frac{1}{K}\frac{\ln(u\ln t)}{u\ln t} + O(1) \tag{5.68}$$

$\ln x/x \leqslant 1/e$ for all $x > 0$, so

$$\phi_\varepsilon(K\varepsilon t) \geqslant -\ln t + \frac{1}{2}\ln\ln t + \frac{1}{2}\ln u + \frac{1}{Ku} + O(1) \tag{5.69}$$

We then see that $\frac{1}{2}\ln u + \frac{1}{Ku} \geqslant \frac{1}{2}\ln\frac{2}{K} + \frac{1}{2} = O(1)$.
So finally $\frac{1}{t_{\mathcal{E}}}\mathcal{E}(z^t|\varnothing) \geqslant -\ln t + \frac{1}{2}\ln\ln t + O(1)$, which is what we wanted. □

Corollary 5.20 (Lower bound on regret for Context Tree Weighting)**.** *Let $K_f \in \mathbb{R}^+$, let f be a K_f-Lipschitz function on $[0,1]$.*

Let $\mathbf{T}$ be a proper (possibly infinite) weighted expert tree with $\mathrm{Dom}\mathcal{E}_\varepsilon = [0,1]$ and such that for all $s \in \mathbf{T}$, $\mathrm{Dom}(\mathcal{E}_s)$ is an interval, with $\lambda(\mathcal{E}_s) = |\mathrm{Dom}(\mathcal{E}_s)|^2$. We have, for any upper well balanced sequence (x_n):

$$-\ln \mathrm{CTW}_{\mathbf{T}}(z^t|\varnothing) \geqslant -t\ln t + \frac{1}{2}t\ln\ln t + O(t), \tag{5.70}$$

where $z_k = (x_k, f(x_k))$.

Proof. By definition, we have $\mathrm{CTW}_{\mathbf{T}} = \mathrm{Bayes}_T(\mathcal{E}_T, w_T)$, so $-\ln \mathrm{CTW}_{\mathbf{T}}(z^t|\varnothing) \geqslant -\max_T \ln \mathcal{E}_T(z^t|\varnothing)$.

Now, for any T, $\mathcal{E}_T = \bigcup_{s \text{ leaf of } \mathbf{T}} \mathcal{E}_s$, so by Lemma 4.21, we have:

$$\mathcal{E}_T(z^t|\varnothing) = \prod_{s \text{ leaf of } T} \mathcal{E}_s(z^t|\varnothing), \tag{5.71}$$

so

$$-\ln \mathcal{E}_T(z^t|\varnothing) \geqslant \sum_{s \text{ leaf of } T} -\ln \mathcal{E}_s(z^t|\varnothing). \tag{5.72}$$

Now, by Theorem 5.19:

$$-\ln \mathcal{E}_T(z^t|\varnothing) \geqslant \sum_{s \text{ leaf of } T} -t_{\mathcal{E}_s}\ln t + \frac{1}{2}t_{\mathcal{E}_s}\ln\ln t + O(t_{\mathcal{E}_s}), \tag{5.73}$$

where $O(t_{\mathcal{E}_s})$ does not depend on the experts, and, since $\sum_s t_{\mathcal{E}_s} = t$,

$$-\ln \mathcal{E}_T(z^t|\varnothing) \geqslant -t\ln t + \frac{1}{2}t\ln\ln t + O(t). \tag{5.74}$$

So finally,

$$-\ln \mathrm{CTW}_{\mathbf{T}}(z^t|\varnothing) \geqslant -\max_T \ln \mathcal{E}_T(z^t|\varnothing) \geqslant -t\ln t + \frac{1}{2}t\ln\ln t + O(t), \tag{5.75}$$

which is what we wanted. □

Notice in particular that we use intervals of size $\ln t/t$ *even if f is constant*[5], because the regularisation is ε^2: going deeper allows the Normal-Gamma experts to have less added artificial variance.

More generally, since Theorem 5.19 bounds uniformly the loss of individual Normal-Gamma experts, we have the following, stronger result, which applies in particular to Bayesian histograms.

[5]The optimal $u = \frac{\varepsilon t}{\ln t}$ in the proof of Theorem 5.19 is constant.

Corollary 5.21 (Lower bound on regret for Bayesian histograms)**.** *Let $K_f \in \mathbb{R}^+$, let f be a K_f-Lipschitz function on $[0,1]$.*

Let $\mathcal{H}$ be a (possibly infinite) Bayesian combination of (possibly countable) unions of Normal-Gamma experts satisfying condition 5.8 with $\lambda(\mathcal{E}) = |\mathrm{Dom}(\mathcal{E})|^2$, and with $\mathrm{Dom}\mathcal{H} = [0,1]$.

We have, for any upper well balanced sequence (x_n):

$$-\ln \mathcal{H}(z^t|\varnothing) \geqslant -t\ln t + \frac{1}{2}t\ln\ln t + O(t), \tag{5.76}$$

where $z_k = (x_k, f(x_k))$.

In particular, (5.76) holds if $\mathcal{H}$ is an histogram or a Bayesian combination of histograms.

Proof. Same proof as Corollary 5.20. □

This bound does not hold when using *time-dependent* histograms (for example, with $t^{1/3}$ bins): using more complex histograms when more data is available is exactly the switch strategy. The advantage of CTS is that it simultaneously considers many different sequences of histograms, so one of them uses the "right" bin size.

Chapter 6

Numerical experiments

We conclude this part with some numerical experiments, to test some of the new algorithms described here. In particular, we are interested in knowing if CTS is also better than CTW for "reasonably small" t.

We used finite binary trees for the experiments (depth 17 for regression, 30 for text compression). The weighting algorithms used equal to $\frac{1}{2}$, and the switch algorithms used correspond to Definition 4.16, with π_K uniform, freezing rate $\theta = \frac{1}{2}$, and the probability distribution $\pi_\tau : n \mapsto \frac{\ln 2}{n+2} - \frac{\ln 2}{n+1}$ on $\mathbb{N}^*$.

The details specific to the application are given in the corresponding parts.

6.1 Regression

In order to illustrate the results in Section 5.3, we test the CTW and CTW algorithms, and their edge counterparts with Normal-Gamma experts on several test functions. The regularization for the experts was $\alpha = .5$, $\lambda(\varepsilon) = \varepsilon^2$, and $\beta = \frac{\lambda}{\lambda+1}$.

The choice of the log-loss could seem unorthodox, but it is quite natural in the MDL context: we want to minimize the description length of the data, which is $-\ln P(z^n)$. Moreover, minimizing the log-loss amounts to minimizing the square error, assuming Gaussian noise with fixed variance.

Since the (z_i) are real numbers, P is usually a probability density on $\mathbb{R}$, so strictly speaking, $-\ln P(z^n)$ is not the number of bits needed to encode z_k (which is infinite). What we are interested in is encoding the z_i *up to a precision* ε. And in that case, the cost of encoding z_i as "a point in an interval I of size ε" is: $-\ln \int_I P(z)\mathrm{d}z \approx -\ln \varepsilon - \ln P(z_i)$.

We use some classical test functions from [MAJ+98]].

As we can see, in the noiselesse case, the edgewise versions of the algo-

Figure 6.1: Regression functions list

Constant	$x \mapsto 0$	"CTW-friendly" setting.
LargeConstant	$x \mapsto 10^6$	Since CTW has to learn the constant at each step, we can expect CTS to be much better than CTW here
SteepLinear	$x \mapsto 100x$	
SteepLinearConstant	$x \mapsto 100x + 10^6$	
Blocks	$x \mapsto \sum h_j K(x - x_j)$	Should behave like Constant for CTW. $K(x) = \frac{1+\text{sg}(x)}{2}$. x_j, h_j in Figure 6.2
Bumps	$x \mapsto \sum h_j K(\frac{x - x_j}{w_j})$	$K(x) = (1 + \lvert x \rvert)^{-4}$. x_j, h_j, w_j in Figure 6.2
Doppler	$x \mapsto \sqrt{x(1-x)}\sin(\frac{2.1\pi}{x + .05})$	
Wave	$x \mapsto .5 + .2\cos(4\pi x) + .1\cos(24\pi x)$	
Additive noise on f	$x \mapsto f(x) + \varepsilon$	$E(\varepsilon) = 0$.
Multiplicative noise on f	$x \mapsto f(x) + \varepsilon f(x)$	$E(\varepsilon) = 0$. The fact that Normal-Gamma experts adapt their variance should be put to use in this case.

Figure 6.2: Complement to Figure 6.1: Parameters for the Blocks and Bumps functions

Blocks and Bumps, x_j	.1	.13	.15	.23	.25	.40	.44	.65	.76	.78	.81
Blocks, h_j	4	−5	3	−4	5	−4.2	2.1	4.3	−3.1	5.1	−4.2
Bumps, h_j	4	5	3	4	5	4.2	2.1	4.3	3.1	5.1	4.2
Bumps, w_j	.005	.005	.006	.01	.01	.03	.01	.01	.005	.008	.005

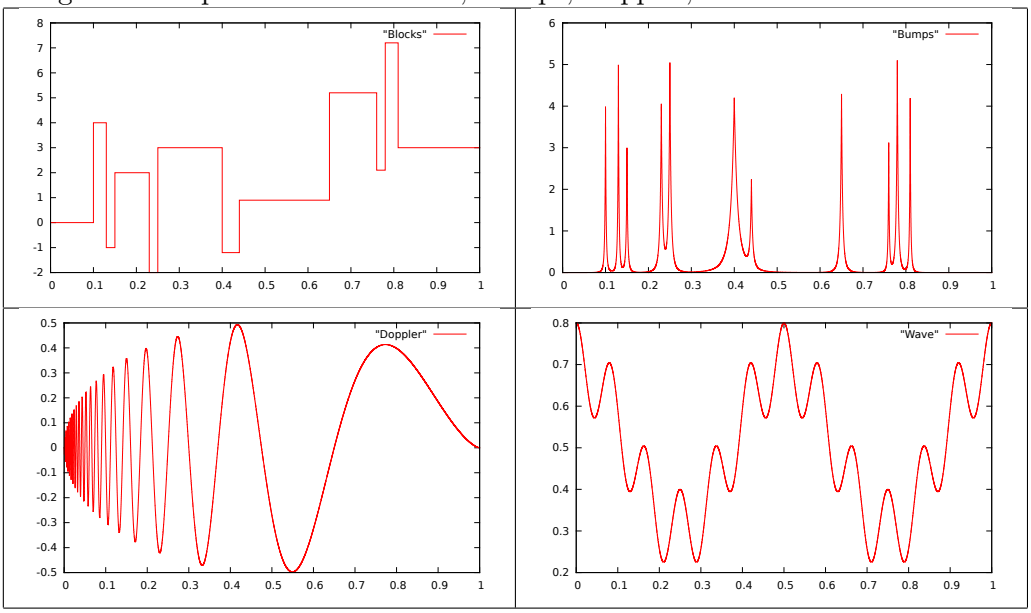
rithm are systematically superior to the non edgewise versions (except for the CTW algorithms on SteepLinearConstant), and switching is systematically superior to weighting (except for a tie on the non edgewise versions on SteepLinearConstant), which confirms our intuition.

However, the performance of the switching algorithms on LargeConstant and SteepLinearConstant is surprising: we would have expected a performance similar to the Constant and SteepLinear functions, since Switch should have paid the price of fixing the constant only once.

The second set of experiments consisted in comparing these algorithms on noisy functions with an oracle (in the noiseless case for density, the loss incurred by an oracle would be $-\infty$: we know $f(x)$ so for any ε, the cost of encoding $f(x)$ up to ε is zero. Intuitively, the "density of a Dirac" is infinite).

In the noisy case, the differences between the algorithms are smaller. In particular, the difference between the edgewise and the non edgewise version of an algorithm is extremely small, with a very slight edge for the

Figure 6.3: Complement to Figure 6.1: Graphs of the functions. From left to right then top to bottom: Blocks, Bumps, Doppler, Wave.



	CTW	CTS	ECTW	ECTS
Constant	−12.207	−12.848	−12.551	−13.160
LargeConstant	8.184	6.586	7.729	6.271
SteepLinear	−4.955	−6.375	−5.266	−6.450
SteepLinearConstant	6.586	6.586	7,730	6.272
Blocks	−11.205	−11.925	−11.560	−12.239
Bumps	−9.710	−10.427	−9.931	−10.559
Doppler	−9.286	−10.051	−9.424	−10.062
Wave	−10.172	−10.987	−10.343	−11.030

Figure 6.4: Average loss in bits after 2.10^5 points in the noiseless case. Average on ten runs.

	CTW	CTS	ECTW	ECTS	Oracle
AddDoppler	−1.992	−2.024	−1.991	−2.020	$-\log_2(\frac{1}{0,2}) \approx -2.322$
AddWave	−2.032	−2.047	−2.032	−2.044	$-\log_2(\frac{1}{0.2}) \approx -2.322$
MultDoppler	−4.198	−4.296	−4.196	−4.279	−4.788
MultWave	−3.091	−3.113	−3.089	−3.109	−3.403

Figure 6.5: Average loss in bits after 2.10^5 points in the noisy case. Average on ten runs. For additive and multiplicative noise, $\varepsilon \sim \text{Unif}[-.1, .1]$.

non edgewise algorithms[1]. CTS remains consistently better than CTW, though, and the average performance of the algorithms is not very far from the oracle.

[1]The samples have very small variance, so the difference is actually significant for CTS and ECTS.

6.2 Text Compression

We test different algorithms on the Calgary corpus, with KT estimators at the leaves. We expect the results of the "new" algorithms to be comparable with those of CTS described in [VNHB11], since the only difference is the choice of the switch distribution. This point of view is developed in Section 6.2.1.

Figure 6.6: Performance of different expert tree algorithms at depth 30 on the Calgary corpus. The results of CTW_{48} and CTS_{48} are taken from [VNHB11] (non enhanced versions with depth 48). In bold: best performance *among the first four algorithms.*)

	bib	book1	book2	geo	news	obj1	obj2	paper1	paper2	paper3	paper4	paper5	paper6	pic	progc	progl	progp	trans
CTW	**2.30**	**2.40**	2.26	5.02	**2.85**	4.64	**3.30**	**2.86**	**2.61**	**2.98**	**3.50**	**3.73**	**3.00**	0.91	**3.01**	2.16	**2.25**	**2.15**
CTS	2.32	2.41	**2.25**	**5.01**	**2.85**	**4.57**	3.34	2.88	2.62	3.00	3.52	3.75	3.02	0.91	3.02	**2.15**	2.26	2.17
ECTW	2.38	2.42	2.29	5.06	2.90	4.67	3.33	2.95	2.67	3.07	3.61	3.83	3.09	0.91	3.11	2.21	2.33	2.25
ECTS	2.41	2.43	2.28	5.05	3.00	4.64	4.63	2.97	2.69	3.09	3.63	3.86	3.12	**0.90**	3.13	2.28	2.35	2.28
CTW_{48}	2.25	2.31	2.12	5.01	2.78	4.63	3.19	2.84	2.59	2.97	3.50	3.73	2.99	0.90	3.00	2.11	2.24	2.09
CTS_{48}, from [VNHB11]	2.23	2.32	2.10	5.05	2.77	4.70	3.16	2.78	2.56	2.95	3.48	3.70	2.93	0.91	2.94	2.05	2.12	1.95

As we can see, most of the time, vanilla CTW gives the best performance, although our version of CTS is always very close (never more than .02 bits per byte in favor of CTW). Interestingly, there is apparently no correlation between the files where our implementation of CTS is better than CTW and the files where CTS from [VNHB11] is better than CTW.

6.2.1 CTS in [VNHB11]

As discussed before, a Context Tree Switching algorithm, using different switch distributions, had already been defined in [VNHB11].

The Markov model corresponding to these distributions was the one described in Section 4.2.3 with two differences:

- The switch in [VNHB11] never freezes.
- The switching probabilities (σ_t) are allowed to depend on the global time T of the algorithm.[2]

It was noticed that choosing $\sigma_t = \frac{1}{t}$ (so t is the "local" time) for the $\mathcal{E}_s$ gave poor empirical performance, contrary to $\sigma_t = \frac{1}{T}$ (where $T(t,s) := \min\{u \in \mathbb{N} || x^u \cap \text{Dom}(\mathcal{E}_s)| \geqslant t\}$ is the global time, if the data are the (x_i)). Moreover, a theoretical bound on CTS regret was obtained. We believe that an intuitive reason for this is the following:

[2] In particular, these experts do not satisfy the "no peeking out condition". However, the CTS update is still easy to compute, since the global time is the same for all experts by definition.

For context trees, the situation is very similar to the one described in the introduction of [vEGdR12], comparing a Markov model of order 1 and a Markov model of order 2 for text compression: we can expect that at some point, any given node will have enough information to "open"[3], and once it is open (i.e. once the local experts at its children nodes have enough data to predict well), it should stay open (the children nodes should continue being better than their father).

However, the typical sequence predicted by a switch distribution that does not freeze with a switching rate $\sigma_t = \frac{1}{t}$ switches infinitely many times (since $\sum_t \frac{1}{t}$ diverges). Choosing $\sigma_t = \frac{1}{T}$ instead of $\sigma_t = \frac{1}{t}$ allows for less frequent switching when t is large in deep nodes, which could partly explain the improved performance.

This property is also probably the reason why our implementation of CTS and CTS from [VNHB11] behave differently: the latter is better suited for models that change over time (as suggested by Theorem 3 in [VNHB11]), while the former should be better on a fixed model. More investigation is needed to confirm this intuition.

[3]Supposing that the best model is not the local expert $\mathcal{E}_s$.

Part III

Geodesic Information Geometric Optimization

Consider an objective function $f\colon X \to \mathbb{R}$ to be minimized. We suppose we have absolutely no knowledge about f: the only thing we can do is ask for its value at any point $x \in X$ (black-box optimization) and that the evaluation of f is a costly operation. We are going to study algorithms that can be described in the IGO framework (see [OAAH11]).

We consider the following optimization procedure:

We choose $(P_\theta)_{\theta\in\Theta}$ a family of probability distributions (which will be given a Riemannian manifold structure, following [AN07]) on X and an initial probability distribution P_{θ^0}. Now, we replace f by $F\colon \Theta \to \mathbb{R}$ (for example $F(\theta) = E_{x\sim P_\theta}[f(x)]$), and we optimize F by gradient descent, corresponding to the gradient flow:

$$\frac{\mathrm{d}\theta^t}{\mathrm{d}t} = -\nabla_\theta E_{x\sim P_\theta}[f(x)]. \tag{6.1}$$

However, because of the gradient, this equation depends entirely on the parametrization we chose for Θ, which is disturbing: we do not want to have two different updates, because we chose different parameters to represent the objects with which we are working. Moreover, in the case of a function with several local minima, changing the parametrization can change the attained optimum (see [MP14], for example). That is why invariance is a design principle behind IGO. More precisely, we want invariance with respect to monotone transformations of f and invariance under reparametrization of Θ.

The IGO framework uses the geometry of the family Θ, which is given by the Fisher metric to provide a differential equation on θ with the desired properties, but because of the discretization of time needed to obtain an explicit algorithm, we lose invariance under reparametrization of θ: two IGO algorithms applied to the same function to be optimized, but with different parametrizations, coincide only at first order in the step size. A possible solution to this problem is geodesic IGO (GIGO), introduced here (see also IGO-Maximum Likelihoodin [OAAH11], for example.): the initial direction of the update at each step of the algorithm remains the same as in IGO, but instead of moving straight for the chosen parametrization, we use the Riemannian manifold structure of our family of probability distributions (see [AN07]) by following its geodesics.

Finding the geodesics of a Riemannian manifold is not always easy, but Noether's theorem will allow us to obtain quantities that are preserved along the geodesics, thus allowing, in the case of Gaussian distributions, one to obtain a first order differential equation satisfied by the geodesics, which makes their computation easier.

Although the geodesic IGO algorithm is not, strictly speaking, parametrization-invariant when no closed form for the geodesics is known, it is possible to compute them at arbitrary precision without increasing the numbers of objective function calls.

Chapter 7 consists in preliminaries: we recall the IGO algorithm, introduced in [OAAH11], and we describe particular cases of the xNES and CMA-ES updates as IGO updates.

Chapter 8 focuses on Noether's theorem: in Section 8.1, after a reminder about Riemannian geometry, we state Noether's theorem, which will be our main tool to compute the GIGO update for Gaussian distributions. In Section 8.2, we consider Gaussian distributions with a covariance matrix proportional to the identity matrix: this space is isometric to the hyperbolic space, and the geodesics of the latter are known. Finally, in Section 8.3, we consider the general Gaussian case, and we use Noether's theorem to obtain two different sets of equations to compute the GIGO update. The equations are known (see [Eri87, CO91, ITW11]), but the connection with Noether's theorem has not been mentioned.[4] We then give the explicit solution for these equations, from [CO91].

In chapter 9, we introduce two possible slight modifications of the IGO algorithm to incorporate the direction-dependent learning rates used in CMA-ES and xNES. We then compare these different algorithms and prove that while xNES is not GIGO in general, it can be described as "Blockwise GIGO", thus recovering xNES from abstract principles.

Finally, Chapter 10 presents numerical experiments, which suggest that when using GIGO with Gaussian distributions, the step size must be chosen carefully.

[4] Another interesting point is that Noether's theorem shortcuts the tedious computation of the Christoffel symbols.

Chapter 7

The IGO framework

In this section, we recall what the IGO framework is and we define the geodesic IGO update. Consider again Equation (6.1):

$$\frac{\mathrm{d}\theta^t}{\mathrm{d}t} = -\nabla_\theta E_{x\sim P_\theta}[f(x)].$$

As we just saw:

- The gradient depends on the parametrization of our space of probability distributions (see 7.3 for an example).
- The equation is not invariant under monotone transformations of f. For example, the optimization for $10f$ moves ten times faster than the optimization for f.

In this section, we recall how IGO deals with this (see [OAAH11] for a better presentation).

7.1 Invariance under Reparametrization of θ: Fisher Metric

In order to achieve invariance under reparametrization of θ, it is possible to turn our family of probability distributions into a Riemannian manifold (this is the main topic of information geometry; see [AN07]), which allows us to use a canonical, parametrization-invariant gradient (called the natural gradient). The first brick in this construction is the differential Kullback–Leibler divergence:

Definition 7.1. *Let P, Q be two probability distributions on X. The Kullback–Leibler divergence of Q from P is defined by:*

$$\mathrm{KL}(Q\|P) = \int_X \ln(\frac{Q(x)}{P(x)})\mathrm{d}Q(x). \tag{7.1}$$

It is the immediate generalization of the Kullback–Leibler divergence in Definition 2.5, and by definition, it does not depend on the parametrization. It is not symmetrical, but if for all x, the application $\theta \mapsto P_\theta(x)$ is C^2, then a second-order expansion yields:

$$\mathrm{KL}(P_{\theta+\mathrm{d}\theta} \| P_\theta) = \frac{1}{2} \sum_{i,j} I_{ij}(\theta) \mathrm{d}\theta_i \mathrm{d}\theta_j + o(\mathrm{d}\theta^2), \tag{7.2}$$

where:

$$I_{ij}(\theta) = \int_X \frac{\partial \ln P_\theta(x)}{\partial \theta_i} \frac{\partial \ln P_\theta(x)}{\partial \theta_j} \mathrm{d}P_\theta(x) = -\int_X \frac{\partial^2 \ln P_\theta(x)}{\partial \theta_i \partial \theta_j} \mathrm{d}P_\theta(x). \tag{7.3}$$

This is enough to endow the family $(P_\theta)_{\theta \in \Theta}$ with a Riemannian manifold structure: a Riemannian manifold M is a differentiable manifold, which can be seen as pieces of $\mathbb{R}^n$ glued together, with a metric . The metric at x is a symmetric positive-definite quadratic form on the tangent space of M at x: it indicates how expensive it is to move in a given direction on the manifold. We will think of the updates of the algorithms that we will be studying as paths on M.

The matrix $I(\theta)$ is called the "Fisher information matrix", and the metric it defines is called the "Fisher metric".

Given a metric, it is possible to define a gradient attached to this metric; the key property of the gradient is that for any smooth function f:

$$f(x+h) = f(x) + \sum_i h_i \frac{\partial f}{\partial x_i} + o(\|h\|) = f(x) + \langle h, \nabla f(x) \rangle + o(\|h\|), \tag{7.4}$$

where $\langle x, y \rangle = x^T I y$ is the dot product in metric I. Therefore, in order to keep the property of Equation (7.4), we must have $\nabla f = I^{-1} \frac{\partial f}{\partial x}$.

We have therefore the following gradient (called the "natural gradient"; see [AN07]):

$$\tilde{\nabla}_\theta = I^{-1}(\theta) \frac{\partial}{\partial \theta}, \tag{7.5}$$

and since the Kullback–Leibler divergence does not depend on the parametrization, neither does the natural gradient.

Later in this paper, we will study families of Gaussian distributions. The following proposition gives the Fisher metric for these families.

Proposition 7.2. *Let $(P_\theta)_{\theta \in \Theta}$ be a family of normal probability distributions: $P_\theta = \mathcal{N}(\mu(\theta), \Sigma(\theta))$. If μ and Σ are C^1, the Fisher metric is given by:*

$$I_{i,j}(\theta) = \frac{\partial \mu^T}{\partial \theta_i} \Sigma^{-1} \frac{\partial \mu}{\partial \theta_j} + \frac{1}{2} \operatorname{tr}\left(\Sigma^{-1} \frac{\partial \Sigma}{\partial \theta_i} \Sigma^{-1} \frac{\partial \Sigma}{\partial \theta_j} \right). \tag{7.6}$$

Proof: This is a non-trivial calculation. See [SSUSCU81] or [PF86] for more details.

As we will often be working with Gaussian distributions, we introduce the following notation:

Notation 7.3. *$\mathbb{G}_d$ is the manifold of Gaussian distributions in dimension d, equipped with the Fisher metric. $\tilde{\mathbb{G}}_d$ is the manifold of Gaussian distributions in dimension d, with the covariance matrix proportional to identity in the canonical basis of $\mathbb{R}^d$, equipped with the Fisher metric.*

7.2 IGO Flow, IGO Algorithm

In IGO [OAAH11], invariance with respect to monotone transformations is achieved by replacing f by the following transform; we set:

$$q(x) = P_{x' \sim P_\theta}(f(x') \leqslant f(x)), \tag{7.7}$$

a non-increasing function $w\colon [0;1] \to \mathbb{R}$ is chosen (the selection scheme), and finally, $W_\theta^f(x) = w(q(x))$ (this definition has to be slightly changed if the probability of a tie is not zero, see [OAAH11] for more details). By performing a gradient descent on $E_{x\sim P_\theta}[W_{\theta^t}^f(x)]$, we obtain the "IGO flow":

$$\frac{\mathrm{d}\theta^t}{\mathrm{d}t} = \tilde{\nabla}_\theta \int_X W_{\theta^t}^f(x) P_\theta(\mathrm{d}x) = \int_X W_{\theta^t}^f(x) \tilde{\nabla}_\theta \ln P_\theta(x) P_{\theta^t}(\mathrm{d}x). \tag{7.8}$$

Notice that the function we are optimizing is $E_{x\sim P_\theta}[W_{\theta^t}^f(x)]$ and not $E_{x\sim P_\theta}[W_\theta^f(x)]$ (the second function is constant and always equal to $\int_0^1 w$). In particular, the function for which we are performing the gradient descent changes at each step, although their optimum (a Dirac at the minimum of f) does not: the IGO flow is not a gradient flow; it is simply a vector flow given by the gradient of interrelated functions.

For practical implementation, the integral in (7.8) has to be approximated. For the integral itself, the Monte-Carlo method is used; N values $(x_1, ..., x_N)$ are sampled from the distribution P_{θ^t}, and the integral becomes:

$$\frac{1}{N} \sum_{i=1}^{N} W_{\theta^t}^f(x_i) \tilde{\nabla}_\theta \ln P_\theta(x_i) \tag{7.9}$$

and we approximate $\frac{1}{N} W_\theta^f(x_i) = \frac{1}{N} w(q(x_i))$ by $\hat{w}_i = \frac{1}{N} w(\frac{\mathrm{rk}(x_i)+1/2}{N})$, where $\mathrm{rk}(x_i) = |\{j, f(x_j) < f(x_i)\}|$: it can be proven (see [OAAH11]) that $\lim_{N\to\infty} N\hat{w}_i = W_f^{\theta^t}(x_i)$ (here again, we are assuming that there are no ties).

We now have an algorithm that can be used in practice if the Fisher information matrix is known.

Definition 7.4. *The IGO update associated with parametrization θ, sample size N, step size δt and selection scheme w is given by the following update rule:*

$$\theta^{t+\delta t} = \theta^t + \delta t I^{-1}(\theta^t) \sum_{i=1}^{N} \hat{w}_i \frac{\partial \ln P_\theta(x_i)}{\partial \theta}. \tag{7.10}$$

We call IGO speed the vector $I^{-1}(\theta^t) \sum_{i=1}^{N} \hat{w}_i \frac{\partial \ln P_\theta(x_i)}{\partial \theta}$.

Notice that one could start directly with the $\hat{w}_i$ rather than w, as we will do later.

Replacing f by its expected value under a probability distribution P_θ and optimizing over θ using the natural gradient have already been discussed. For example, in the case of a function f defined on $\{0,1\}^n$, IGO with the Bernoulli distributions yields the algorithm, PBIL[BC95]. Another similar approach (stochastic relaxation) is given in [MMP11]. For a continuous function, as we will see later, the IGO framework recovers several known ranked-based natural gradient algorithms, such as pure rank-μ CMA-ES [KMH+03], xNES or SNES (Separable Natural Evolution Strategies) [WSG+14]. See [Hua13] or [AMS08] for other, not necessarily gradient-based, optimization algorithms on manifolds.

7.3 Geodesic IGO

Although the IGO flow associated with a family of probability distributions is intrinsic (it only depends on the family itself, not the parametrization we choose for it), the IGO update is not. However, the difference between two steps of IGO that differ only by the parametrization is only $O(\delta t^2)$, whereas the different between two vanilla gradient descents with different parametrizations is $O(\delta t)$.

Intuitively, the reason for this difference is that two IGO algorithms start at the same point and follow "straight lines" with the same initial speed, but the definition of "straight lines" changes with the parametrization.

For instance, in the case of Gaussian distributions, let us consider two different IGO updates with Gaussian distributions in dimension one, the first one with parametrization (μ, σ) and the second one with parametrization $(\mu, c := \sigma^2)$. We suppose that the IGO speed for the first algorithm is $(\dot{\mu}, \dot{\sigma})$. The corresponding IGO speed in the second parametrization is given by the identity $\dot{c} = 2\sigma\dot{\sigma}$. Therefore, the first algorithm gives the standard deviation $\sigma_{\text{new},1} = \sigma_{\text{old}} + \delta t \dot{\sigma}$ and the variance $c_{\text{new},1} = (\sigma_{\text{new},1})^2 = c_{\text{old}} + 2\delta t \sigma_{\text{old}} \dot{\sigma} + \delta t^2 \dot{\sigma}^2 = c_{\text{new},2} + \delta t^2 \dot{\sigma}^2$.

The geodesics of a Riemannian manifold are the generalization of the notion of a straight line: they are curves that locally minimize length. In particular, given two points a and b on the Riemannian manifold M, the shortest path from a to b is always a geodesic (the converse is not true,

though). The notion will be explained precisely in Section 8.1, but let us define the geodesic IGO algorithm, which follows the geodesics of the manifold instead of following the straight lines for an arbitrary parametrization.

Definition 7.5 (GIGO)**.** *The geodesic IGO update (GIGO) associated with sample size N, step size δt and selection scheme w is given by the following update rule:*

$$\theta^{t+\delta t} = \exp_{\theta^t}(Y\delta t) \tag{7.11}$$

where:

$$Y = I^{-1}(\theta^t)\sum_{i=1}^{N} \hat{w}_i \frac{\partial \ln P_\theta(x_i)}{\partial \theta}, \tag{7.12}$$

is the IGO speed and $\exp_{\theta^t}$ is the exponential of the Riemannian manifold Θ. Namely, $\exp_{\theta^t}(Y\delta t)$ is the endpoint of the geodesic of Θ starting at θ^t, with initial speed Y, after a time δt. By definition, this update does not depend on the parametrization θ.

Notice that while the GIGO update is compatible with the IGO flow (in the sense that when $\delta t \to 0$ and $N \to \infty$, a parameter θ^t updated according to the GIGO algorithm is a solution of Equation (7.8), the equation defining the IGO flow), it not necessarily an IGO update. More precisely, the GIGO update is an IGO update if and only if the geodesics of Θ are straight lines for some parametrization (by Beltrami's theorem, this is equivalent to Θ having constant curvature).

This is a particular case of a retraction [AMS08]: a map from the tangent bundle of a manifold to the manifold itself satisfying a rigidity condition. Arguably, the Riemannian exponential is the most natural retraction, since it depends only on the Riemannian manifold itself and not on any decomposition. However, in general, the geodesics are difficult to compute.

In the next section, we will state Noether's theorem, which will be our main tool to compute the GIGO update for Gaussian distributions.

7.4 Comparable pre-existing algorithms

In this section, we recall the xNES and pure rank-μ CMA-ES, and we describe them in the IGO framework, thus allowing a reasonable comparison with the GIGO algorithms.

7.4.1 xNES

We recall a restriction of the xNES algorithm, introduced in [GSY+10] (this restriction is sufficient to describe the numerical experiments in [GSY+10]).

Definition 7.6 (xNES algorithm)**.** *The xNES algorithm with sample size N, weights w_i and learning rates η_μ and η_Σ updates the parameters $\mu \in \mathbb{R}^d$, $A \in M_d(\mathbb{R})$ with the following rule: At each step, N points $x_1, ..., x_N$ are sampled from the distribution $\mathcal{N}(\mu, AA^T)$. Without loss of generality, we assume $f(x_1) < ... < f(x_N)$. The parameter is updated according to:*

$$\mu \leftarrow \mu + \eta_\mu A G_\mu,$$

$$A \leftarrow A \exp(\eta_\Sigma G_M/2),$$

where, setting $z_i = A^{-1}(x_i - \mu)$:

$$G_\mu = \sum_{i=1}^{N} w_i z_i,$$

$$G_M = \sum_{i=1}^{N} w_i (z_i z_i^T - I).$$

The more general version decomposes the matrix A as σB, where $\det B = 1$, and uses two different learning rates for σ and for B. We gave the version where these two learning rates are equal (in particular, for the default parameters in [GSY+10], these two learning rates are equal). This restriction of the xNES algorithm can be described in the IGO framework, provided all of the learning rates are equal (most of the elements of the proof can be found in [GSY+10] (the proposition below essentially states that xNES is a natural gradient update) or in [OAAH11]):

Proposition 7.7 (xNES as IGO)**.** *The xNES algorithm with sample size N, weights w_i and learning rates $\eta_\mu = \eta_\Sigma = \delta t$ coincides with the IGO algorithm with sample size N, weights w_i, step size δt and in which, given the current position (μ_t, A_t), the set of Gaussians is parametrized by:*

$$\phi_{\mu_t, A_t} : (\delta, M) \mapsto \mathcal{N}\left(\mu_t + A_t\delta, \left(A_t \exp(\frac{1}{2}M)\right)\left(A_t \exp(\frac{1}{2}M)\right)^T\right),$$

with $\delta \in \mathbb{R}^m$ and $M \in \mathrm{Sym}(\mathbb{R}^m)$.

The parameters maintained by the algorithm are (μ, A), and the x_i are sampled from $\mathcal{N}(\mu, AA^T)$.

Proof. Let us compute the IGO update in the parametrization ϕ_{μ_t, A_t}: we have $\delta^t = 0$, $M^t = 0$, and by using Proposition 7.2, we can see that for this parametrization, the Fisher information matrix at $(0, 0)$ is the identity matrix. The IGO update is therefore,

$$(\delta, M)^{t+\delta t} = (\delta, M)^t + \delta t Y_\delta(\delta, M) + \delta t Y_M(\delta, M) = \delta t Y_\delta(\delta, M) + \delta t Y_M(\delta, M),$$

where:

$$Y_\delta(\delta, M) = \sum_{i=1}^{N} w_i \nabla_\delta \ln(p(x_i|(\delta, M))$$

and:

$$Y_M(\delta, M) = \sum_{i=1}^{N} w_i \nabla_M \ln(p(x_i|(\delta, M)).$$

Since $\mathrm{tr}(M) = \log(\det(\exp(M)))$, we have:

$$\ln P_{\delta,M}(x) = -\frac{d}{2}\ln(2\pi) - \ln(\det A) - \frac{1}{2}\,\mathrm{tr}\, M - \frac{1}{2}\|\exp(-\frac{1}{2}M)A^{-1}(x-\mu-A\delta)\|^2,$$

and a straightforward computation yields:

$$Y_\delta(\delta, M) = \sum_{i=1}^{N} w_i z_i = G_\mu,$$

and:

$$Y_M(\delta, M) = \frac{1}{2}\sum_{i=1}^{N} w_i (z_i z_i^T - I) = G_M.$$

Therefore, the IGO update is:

$$\delta(t+\delta t) = \delta(t) + \delta t G_\mu,$$
$$M(t+\delta t) = M(t) + \delta t G_M,$$

or, in terms of mean and covariance matrix:

$$\mu(t+\delta t) = \mu(t) + \delta t A(t) G_\mu$$

$$A(t+\delta t) = A(t)\exp(\delta t G_M/2),$$

or:

$$\Sigma(t+\delta t) = A(t)\exp(\delta t G_M) A(t)^T.$$

This is the xNES update. □

7.4.2 Pure Rank-μ CMA-ES

We now recall the pure rank-μ CMA-ES algorithm. The general CMA-ES algorithm is described in [Han11].

Definition 7.8 (Pure rank-μ CMA-ES algorithm)**.** *The pure rank-μ CMA-ES algorithm with sample size N, weights w_i and learning rates η_μ and η_Σ is defined by the following update rule: At each step, N points $x_1, ..., x_N$ are sampled from the distribution $\mathcal{N}(\mu, \Sigma)$. Without loss of generality, we assume $f(x_1) < ... < f(x_N)$. The parameter is updated according to:*

$$\mu \leftarrow \mu + \eta_\mu \sum_{i=1}^{N} w_i (x_i - \mu),$$

$$\Sigma \leftarrow \Sigma + \eta_\Sigma \sum_{i=1}^{N} w_i ((x_i - \mu)(x_i - \mu)^T - \Sigma).$$

The pure rank-μ CMA-ES can also be described in the IGO framework; see, for example, [ANOK10].

Proposition 7.9 (Pure rank-μ CMA-ES as IGO)**.** *The pure rank-μ CMA-ES algorithm with sample size N, weights w_i and learning rates $\eta_\mu = \eta_\Sigma = \delta t$ coincides with the IGO algorithm with sample size N, weights w_i, step size δt and the parametrization (μ, Σ).*

Chapter 8

Using Noether's theorem to compute geodesics

8.1 Riemannian Geometry, Noether's Theorem

The goal of this section is to state Noether's theorem. See [AVW89] for the proofs and [Bou07] or [JLJ98] for a more detailed presentation. Noether's theorem states that if a system has symmetries, then there are invariants attached to these symmetries. Firstly, we need some definitions.

Definition 8.1 (Motion in a Lagrangian system)**.** *Let M be a differentiable manifold, TM the set of tangent vectors on M (a tangent vector is identified by the point at which it is tangent and a vector in the tangent space) and*

$$\begin{array}{rccc} \mathcal{L} & : & TM & \to & \mathbb{R} \\ & & (q,v) & \mapsto & \mathcal{L}(q,v) \end{array}$$

a differentiable function called the Lagrangian function (in general, it could depend on t). A "motion in the Lagrangian system $(M,\mathcal{L})$ from x to y" is map $\gamma\colon [t_0,t_1]\to M$, such that:

- $\gamma(t_0)=x$
- $\gamma(t_1)=y$
- *γ is a local extremum of the functional:*

$$\Phi(\gamma)=\int_{t_0}^{t_1}\mathcal{L}(\gamma(t),\dot{\gamma}(t))\mathrm{d}t, \tag{8.1}$$

among all curves $c\colon [t_0,t_1]\to M$, such that $c(t_0)=x$, and $c(t_1)=y$.

For example, when (M,g) is a Riemannian manifold, the length of a curve γ between $\gamma(t_0)$ and $\gamma(t_1)$ is:

$$\int_{t_0}^{t_1}\sqrt{g(\dot{\gamma}(t),\dot{\gamma}(t))}\mathrm{d}t. \tag{8.2}$$

The curves that follow the shortest path between two points $x,y\in M$ are therefore the minima γ of the functional (8.2), such that $\gamma(t_0)=x$ and

$\gamma(t_1) = y$, and the corresponding Lagrangian function is $(q, v) \mapsto \sqrt{g(v, v)}$. However, any curve following the shortest trajectory will have minimum length. For example, if $\gamma_1 : [a, b] \to M$ is a curve of the shortest path, so is $\gamma_2 : t \mapsto \gamma_1(t^2)$: these two curves define the same trajectory in M, but they do not travel along this trajectory at the same speed. This leads us to the following definition:

Definition 8.2 (Geodesics)**.** *Let I be an interval of $\mathbb{R}$ and (M, g) be a Riemannian manifold. A curve $\gamma : I \to M$ is called a geodesic if for all $t_0, t_1 \in I$, $\gamma|_{[t_0,t_1]}$ is a motion in the Lagrangian system $(M, \mathcal{L})$ from $\gamma(t_0)$ to $\gamma(t_1)$, where:*

$$\mathcal{L}(\gamma) = \int_{t_0}^{t_1} g(\dot{\gamma}(t), \dot{\gamma}(t))\mathrm{d}t. \tag{8.3}$$

It can be shown (see [Bou07]) that geodesics are curves that locally minimize length, with constant velocity, in the sense that $\frac{dg(\dot{\gamma}(t),(\dot{\gamma}(t))}{dt} = 0$. In particular, given a starting point and a starting speed, the geodesic is unique. This motivates the definition of the exponential of a Riemannian manifold.

Definition 8.3. *Let (M, g) be a Riemannian manifold. We call the exponential of M the application:*

$$\begin{array}{rccc} \exp & : & TM & \to & M \\ & & (x, v) & \mapsto & \exp_x(v), \end{array}$$

such that for any $x \in M$, if γ is the geodesic of M satisfying $\gamma(0) = x$ and $\gamma'(0) = v$, then $\exp_x(v) = \gamma(1)$.

In order to find an extremal of a functional, the most commonly-used result is called the "Euler–Lagrange equations" (see [AVW89], for example); a motion γ in the Lagrangian system $(M, \mathcal{L})$ must satisfy:

$$\frac{\partial \mathcal{L}}{\partial x}(\gamma(t)) - \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}(\dot{\gamma}(t))\right) = 0. \tag{8.4}$$

By applying this equation with the Lagrangian given by (8.3), it is possible to show that the geodesics of a Riemannian manifold follow the "geodesic equations":

$$\ddot{x}^k + \Gamma^k_{ij}\dot{x}^i\dot{x}^j = 0, \tag{8.5}$$

where the

$$\Gamma^k_{ij} = \frac{1}{2}g^{lk}\left(\frac{\partial g_{jl}}{\partial q_i} + \frac{\partial g_{li}}{\partial q_j} - \frac{\partial g_{ij}}{\partial q_l}\right) \tag{8.6}$$

are called "Christoffel symbols" of the metric g. However, these coefficients are tedious (and sometimes difficult) to compute, and (8.5) is a second order differential equation. Noether's theorem will give us a first order equation to compute the geodesics.

Definition 8.4. *Let $h\colon M \to M$, a diffeomorphism. We say that the Lagrangian system $(M, \mathcal{L})$ admits the symmetry h if for any $(q, v) \in TM$,*

$$\mathcal{L}\left(h(q), \mathrm{d}h(v)\right) = \mathcal{L}(q, v), \tag{8.7}$$

where $\mathrm{d}h$ is the differential of h.

If M is clear in the context, we will sometimes say that $\mathcal{L}$ is invariant under h.

An example will be given in the proof of Theorem 8.13.

We can now state Noether's theorem (see, for example, [AVW89]).

Theorem 8.5 (Noether's Theorem)**.** *If the Lagrangian system $(M, \mathcal{L})$ admits the one-parameter group of symmetries $h^s\colon M \to M$, $s \in \mathbb{R}$, then the following quantity remains constant during motions in the system $(M, \mathcal{L})$. Namely,*

$$I(\gamma(t), \dot{\gamma}(t)) = \frac{\partial \mathcal{L}}{\partial v}\left(\frac{\mathrm{d}h^s(\gamma(t))}{\mathrm{d}s}|_{s=0}\right) \tag{8.8}$$

does not depend on t if γ is a motion in $(M, \mathcal{L})$.

Now, we are going to apply this theorem to our problem: computing the geodesics of Riemannian manifolds of Gaussian distributions.

8.2 GIGO in $\tilde{\mathbb{G}}_d$

If we force the covariance matrix to be either diagonal or proportional to the identity matrix, the geodesics have a simple expression that we give below. In the former case, the manifold we are considering is $(\mathbb{G}_1)^d$, and in the latter case, it is $\tilde{\mathbb{G}}_d$.

The geodesics of $(\mathbb{G}_1)^d$ are given by:

Proposition 8.6. *Let M be a Riemannian manifold; let $d \in \mathbb{N}$; let Φ be the Riemannian exponential of M^d; and let ϕ be the Riemannian exponential of M. We have:*

$$\Phi_{(x_1,\ldots,x_n)}((v_1, \ldots, v_n)) = (\phi_{x_1}(v_1), \ldots, \phi_{x_n}(v_n)) \tag{8.9}$$

In particular, knowing the geodesics of $\mathbb{G}_1$ is enough to compute the geodesics of $(\mathbb{G}_1)^d$.

This is true, because a block of the product metric does not depend on variables of the other blocks.

Consequently, a GIGO update with a diagonal covariance matrix with the sample (x_i) is equivalent to d separate one-dimensional GIGO updates using the same samples. Moreover, $\mathbb{G}_1 \cong \tilde{\mathbb{G}}_1$, the geodesics of which are given below.

We will show that $\tilde{\mathbb{G}}_d$ and the "hyperbolic space", of which the geodesics are known, are isometric.

8.2.1 Preliminaries: Poincaré Half-Plane, Hyperbolic Space

In dimension two, the hyperbolic space is called the "hyperbolic plane" or the Poincaré half-plane. We recall its definition:

Definition 8.7 (Poincaré half-plane)**.** *We call the "Poincaré half-plane" the Riemannian manifold:*

$$\mathcal{H} = \{(x, y) \in \mathbb{R}^2,\ y > 0\},$$

with the metric $ds^2 = \frac{dx^2+dy^2}{y^2}$.

We also recall the expression of its geodesics (see, for example, [GHL04]):

Proposition 8.8 (Geodesics of the Poincaré half-plane)**.** *The geodesics of the Poincaré half-plane are exactly the:*

$$t \mapsto (\Re(z(t)), \Im(z(t)))\,,$$

where:

$$z(t) = \frac{aie^{vt} + b}{cie^{vt} + d}, \tag{8.10}$$

with $ad - bc = 1$ *and* $v > 0$.

The geodesics are half-circles perpendicular to the line $y = 0$ and vertical lines, as shown in Figure 8.1 below.

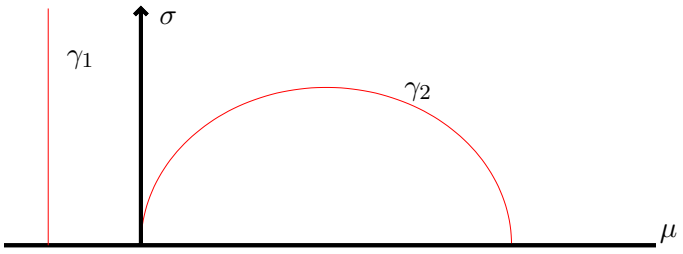


Figure 8.1: Geodesics of the Poincaré half-plane.

The generalization to the higher dimension is the following:

Definition 8.9 (Hyperbolic space)**.** *We call the "hyperbolic space of dimension n" the Riemannian manifold:*

$$\mathcal{H}_n = \{(x_1, ..., x_{n-1}, y) \in \mathbb{R}^n, y > 0\},$$

with the metric $ds^2 = \frac{dx_1^2+...+dx_{n-1}^2+dy^2}{y^2}$ *(or equivalently, the metric given by the matrix* $\mathrm{Diag}(\frac{1}{y^2})$ *).*

The Lagrangian for the geodesics is invariant under all translations along the x_i, so by Noether's theorem, its geodesics stay in a plane containing the direction y and the initial speed . The induced metric on this plane is the metric of the Poincaré half-plane. The geodesics are therefore given by the following proposition:

Proposition 8.10 (Geodesics of the hyperbolic space)**.** *If* $\gamma\colon t \mapsto (x_1(t), ..., x_{n-1}(t), y(t)) = (\boldsymbol{x}(t), y(t))$ *is a geodesic of* $\mathcal{H}_n$*, then there exists* $a, b, c, d \in \mathbb{R}$*, such that* $ad - bc = 1$*, and* $v > 0$*, such that*

$\boldsymbol{x}(t) = \boldsymbol{x}(0) + \frac{\dot{\boldsymbol{x}}_0}{\|\dot{\boldsymbol{x}}_0\|}\tilde{x}(t)$*,* $y(t) = \Im(\gamma_{\mathcal{C}}(t))$*, with* $\tilde{x}(t) = \Re(\gamma_{\mathcal{C}}(t))$ *and:*

$$\gamma_{\mathcal{C}}(t) := \frac{aie^{vt} + b}{cie^{vt} + d}. \tag{8.11}$$

8.2.2 Computing the GIGO Update in $\tilde{\mathbb{G}}_d$

If we want to implement the GIGO algorithm in $\tilde{\mathbb{G}}_d$, we need to compute the natural gradient in $\tilde{\mathbb{G}}_d$ and to be able to compute the Riemannian exponential of $\tilde{\mathbb{G}}_d$.

Using Proposition 7.2, we can compute the metric of $\tilde{\mathbb{G}}_d$ in the parametrization $(\mu, \sigma) \mapsto \mathcal{N}(\mu, \sigma^2 I)$. We find:

$$\begin{pmatrix} \frac{1}{\sigma^2} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \frac{1}{\sigma^2} & 0 \\ 0 & \dots & 0 & \frac{2d}{\sigma^2} \end{pmatrix}. \tag{8.12}$$

Since this matrix is diagonal, it is easy to invert, and we immediately have the natural gradient and, consequently, the IGO speed.

Proposition 8.11. *In* $\tilde{\mathbb{G}}_d$*, the IGO speed* Y *is given by:*

$$Y_\mu = \sum_i \hat{w}_i (x_i - \mu), \tag{8.13}$$

$$Y_\sigma = \sum_i \hat{w}_i \left(\frac{(x_i - \mu)^T (x_i - \mu)}{2d\sigma} - \frac{\sigma}{2} \right). \tag{8.14}$$

Proof. We recall the IGO speed is defined by $Y = I^{-1}(\theta^t) \sum_{i=1}^N \hat{w}_i \frac{\partial \ln P_\theta(x_i)}{\partial \theta}$. Since $P_{\mu,\sigma}(x) = (2\pi\sigma^2)^{-d/2} \exp(-\frac{(x-\mu)^T(x-\mu)}{2\sigma^2})$, we have:

$$\frac{\partial \ln P_{\mu,\sigma}(x)}{\partial \mu} = x - \mu,$$

$$\frac{\partial \ln P_{\mu,\sigma}(x)}{\partial \sigma} = -\frac{d}{\sigma} + \frac{(x-\mu)^T(x-\mu)}{\sigma^3}.$$

The result follows. □

The metric defined by Equation (8.12) is not exactly the metric of the hyperbolic space, but with the substitution $\mu \leftarrow \frac{\mu}{\sqrt{2d}}$, the metric becomes $\frac{2d}{\sigma^2} I$, which is proportional to the metric of the hyperbolic space and, therefore, defines the same geodesics.

Theorem 8.12 (Geodesics of $\tilde{\mathbb{G}}_d$)**.** *If $\gamma \colon t \mapsto \mathcal{N}(\mu(t), \sigma(t)^2 I)$ is a geodesic of $\tilde{\mathbb{G}}_d$, then there exists $a, b, c, d \in \mathbb{R}$, such that $ad - bc = 1$, and $v > 0$, such that:*

$\mu(t) = \mu(0) + \sqrt{2d}\frac{\dot{\mu}_0}{\|\dot{\mu}_0\|}\tilde{r}(t)$, $\sigma(t) = \Im(\gamma_{\mathcal{C}}(t))$, *with* $\tilde{r}(t) = \Re(\gamma_{\mathcal{C}}(t))$ *and*

$$\gamma_{\mathcal{C}}(t) := \frac{aie^{vt} + b}{cie^{vt} + d}. \tag{8.15}$$

Now, in order to implement the corresponding GIGO algorithm, we only need to be able to find the coefficients a, b, c, d, v corresponding to an initial position (μ_0, σ_0) and an initial speed $(\dot{\mu}_0, \dot{\sigma}_0)$. This is a tedious but easy computation, the result of which is given in Proposition B.5.

The pseudocode of GIGO in $\tilde{\mathbb{G}}_d$ is also given in the Appendix: it is obtained by concatenating Algorithms 3 and 9 (Proposition B.5 and the pseudocode in the Appendix allow the metric to be slightly modified; see Section 9.1.1).

8.3 GIGO in $\mathbb{G}_d$

8.3.1 Obtaining a First Order Differential Equation for the Geodesics of $\mathbb{G}_d$

In the case where both the covariance matrix and the mean can vary freely, the equations of the geodesics have been computed in [Eri87] and [CO91]. However, these articles start with the equations of the geodesics obtained with the Christoffel symbols, then partially integrate them . These equations are in fact a consequence of Noether's theorem and can be found directly.

Theorem 8.13. *Let $\gamma : t \mapsto \mathcal{N}(\mu_t, \Sigma_t)$ be a geodesic of $\mathbb{G}_d$. Then, the following quantities do not depend on t:*

$$J_\mu = \Sigma_t^{-1}\dot{\mu}_t, \tag{8.16}$$

$$J_\Sigma = \Sigma_t^{-1}(\dot{\mu}_t\mu_t^T + \dot{\Sigma}_t). \tag{8.17}$$

Proof. This is a direct application of Noether's theorem, with suitable groups of diffeomorphisms. By Proposition 7.2, the Lagrangian associated with the geodesics of $\mathbb{G}_d$ is:

$$\mathcal{L}(\mu, \Sigma, \dot{\mu}, \dot{\Sigma}) = \dot{\mu}^T\Sigma^{-1}\dot{\mu} + \frac{1}{2}\operatorname{tr}(\dot{\Sigma}\Sigma^{-1}\dot{\Sigma}\Sigma^{-1}). \tag{8.18}$$

Its derivative is:

$$\frac{\partial \mathcal{L}}{\partial \dot{\theta}} = \left[(h, H) \mapsto 2\dot{\mu}^T \Sigma^{-1} h + \operatorname{tr}(H\Sigma^{-1}\dot{\Sigma}\Sigma^{-1})\right]. \tag{8.19}$$

Let us show that this Lagrangian is invariant under affine changes of basis (thus illustrating Definition 8.4).

The general form of an affine change of basis is $\phi_{\mu_0,A} : (\mu, \Sigma) \mapsto (A\mu + \mu_0, A\Sigma A^T)$, with $\mu_0 \in \mathbb{R}^d$ and $A \in \mathrm{GL}_d(\mathbb{R})$.

We have:

$$\begin{aligned}\mathcal{L}(\phi_{\mu_0,A}(\mu, \Sigma), d\phi_{\mu_0,A}(\dot{\mu}, \dot{\Sigma})) =& \dot{\overline{A\mu}}^T (A\Sigma A^T)^{-1} \dot{\overline{A\mu}} + \\ & \frac{1}{2} \operatorname{tr}\left(\dot{\overline{A\Sigma A^T}} (A\Sigma A^T)^{-1} \dot{\overline{A\Sigma A^T}} (A\Sigma A^T)^{-1}\right),\end{aligned} \tag{8.20}$$

and since $\dot{\overline{A\mu}} = A\dot{\mu}$ and $\dot{\overline{A\Sigma A^T}} = A\dot{\Sigma}A^T$, we find easily that:

$$\mathcal{L}(\phi_{\mu_0,A}(\mu, \Sigma), d\phi_{\mu_0,A}(\dot{\mu}, \dot{\Sigma})) = \mathcal{L}(\mu, \Sigma, \dot{\mu}, \dot{\Sigma}), \tag{8.21}$$

or in other words: $\mathcal{L}$ is invariant under $\phi_{\mu_0,A}$ for any $\mu_0 \in \mathbb{R}^d, A \in GL_d(\mathbb{R})$.

In order to use Noether's theorem, we also need one-parameter groups of transformations. We choose the following:

1. Translations of the mean vector. For any $i \in [1, d]$, let $h_i^s \colon (\mu, \Sigma) \mapsto (\mu + se_i, \Sigma)$, where e_i is the i-th basis vector. We have $\frac{\mathrm{d}h_i^s}{ds}|_{s=0} = (e_i, 0)$, so by Noether's theorem,

   $$\frac{\partial \mathcal{L}}{\partial \dot{\theta}}(e_i, 0) = 2\dot{\mu}^T \Sigma^{-1} e_i = 2e_i^T \Sigma^{-1} \dot{\mu}$$

   remains constant for all i. The fact that J_μ is an invariant immediately follows.

2. Linear base changes. For any $i, j \in [1, d]$, let $h_{i,j}^s : (\mu, \Sigma) \mapsto (\exp(sE_{ij})\mu, \exp(sE_{ij})\Sigma \exp(sE_{ji}))$, where E_{ij} is the matrix with a one at position (i, j) and zeros elsewhere. We have:

   $$\frac{\mathrm{d}h_{E_{ij}}^s}{\mathrm{d}s}|_{s=0} = (E_{ij}\mu, E_{ij}\Sigma + \Sigma E_{ji}).$$

   Therefore, by Noether's theorem, we then obtain the following invariants:

   $$\begin{aligned} J_{ij} :=& \frac{\partial \mathcal{L}}{\partial \dot{\theta}}(E_{ij}\mu, E_{ij}\Sigma + \Sigma E_{ji}) && (8.22)\\ =& 2\,\dot{\mu}^T \Sigma^{-1} E_{ij}\mu + \operatorname{tr}((E_{ij}\Sigma + \Sigma E_{ji})\Sigma^{-1}\dot{\Sigma}\Sigma^{-1}) && (8.23)\\ =& 2\,(\Sigma^{-1}\dot{\mu})^T E_{ij}\mu + \operatorname{tr}(E_{ij}\dot{\Sigma}\Sigma^{-1}) + \operatorname{tr}(E_{ji}\Sigma^{-1}\dot{\Sigma}) && (8.24)\\ =& 2(J_\mu \mu^T)_{ij} + 2(\Sigma^{-1}\dot{\Sigma})_{ij}, && (8.25)\end{aligned}$$

   and the coefficients of J_Σ in (8.17) are the $(J_{ij}/2)$.

□

This leads us to first order equations satisfied by the geodesics mentioned in [Eri87, CO91, ITW11].

Theorem 8.14 (GIGO-Σ). *$t \mapsto \mathcal{N}(\mu_t, \Sigma_t)$ is a geodesic of $\mathbb{G}_d$ if and only if $\mu : t \mapsto \mu_t$ and $\Sigma : t \mapsto \Sigma_t$ satisfy the equations:*

$$\dot{\mu}_t = \Sigma_t J_\mu \tag{8.26}$$

$$\dot{\Sigma}_t = \Sigma_t (J_\Sigma - J_\mu \mu_t^T) = \Sigma_t J_\Sigma - \dot{\mu}_t \mu_t^T, \tag{8.27}$$

where:

$$J_\mu = \Sigma_0^{-1} \dot{\mu}_0,$$

and:

$$J_\Sigma = \Sigma_0^{-1} \left(\dot{\mu}_0 \mu_0^T + \dot{\Sigma}_0 \right).$$

Proof: This is an immediate consequence of Proposition 8.13.

These equations can be solved analytically (see [CO91]); however, usually, that is not the case, and they have to be solved numerically, for example with the Euler method (the corresponding algorithm, which we call GIGO-Σ, is described in the Appendix). The goal of the remainder of the subsection is to show that having to use the Euler method is fine.

To avoid confusion, we will call the step size of the GIGO algorithm (δt in Proposition 7.5) "GIGO step size" and the step size of the Euler method (inside a step of the GIGO algorithm) "Euler step size".

Having to solve our equations numerically brings two problems:

The first one is a theoretical problem: the main reason to study GIGO is its invariance under reparametrization of θ, and we lose this invariance property when we use the Euler method. However, GIGO can get arbitrarily close to invariance by decreasing the Euler step size. In other words, the difference between two different IGO algorithms is $O(\delta t^2)$, and the difference between two different implementations of the GIGO algorithm is $O(h^2)$, where h is the Euler step size; it is easier to reduce the latter. Still, without a closed form for the geodesics of $\mathbb{G}_d$, the GIGO update is rather expensive to compute, but it can be argued that most of the computation time will still be the computation of the objective function f.

The second problem is purely numerical: we cannot guarantee that the covariance matrix remains positive-definite along the Euler method. Here, apart from finding a closed form for the geodesics, we have two solutions.

We can enforce this *a posteriori*: if the covariance matrix we find is not positive-definite after a GIGO step, we repeat the failed GIGO step with a reduced Euler step size (in our implementation, we divided it by four; see Algorithm 4 in the Appendix.).

The other solution is to obtain differential equations on a square root of the covariance matrix (*any* matrix A, such that $\Sigma = AA^T$).

Theorem 8.15 (GIGO-A). *If $\mu : t \mapsto \mu_t$ and $A : t \mapsto A_t$ satisfy the equations:*

$$\dot{\mu}_t = A_t A_t^T J_\mu, \tag{8.28}$$

$$\dot{A}_t = \frac{1}{2}(J_\Sigma - J_\mu \mu_t^T)^T A_t, \tag{8.29}$$

where:

$$J_\mu = (A_0^{-1})^T A_0^{-1} \mu_0$$

and:

$$J_\Sigma = (A_0^{-1})^T A_0^{-1}(\dot{\mu}_0 \mu_0^T + \dot{A}_0 A_0^T + A_0 \dot{A}_0^T),$$

then $t \mapsto \mathcal{N}(\mu_t, A_t A_t^T)$ is a geodesic of $\mathbb{G}_d$.

Proof. This is a simple rewriting of Theorem 8.14: if we write $\Sigma := AA^T$, we find that J_μ and J_Σ are the same as in Theorem 8.14, and we have:

$$\dot{\mu} = \Sigma J_\mu,$$

and:

$$\dot{\Sigma} = (\dot{A}A^T + A\dot{A}^T) = \frac{1}{2}(J_\Sigma - J_\mu \mu^T)^T AA^T + \frac{1}{2}AA^T(J_\Sigma - J_\mu \mu^T)$$

$$= \frac{1}{2}(J_\Sigma - J_\mu \mu^T)^T \Sigma + \frac{1}{2}\Sigma(J_\Sigma - J_\mu \mu^T) = \frac{1}{2}\Sigma(J_\Sigma - J_\mu \mu^T) + \frac{1}{2}[\Sigma(J_\Sigma - J_\mu \mu^T)]^T.$$

By Theorem 8.14, $\Sigma(J_\Sigma - J_\mu \mu^T)$ is symmetric (since $\dot{\Sigma}$ has to be symmetric). Therefore, we have $\dot{\Sigma} = \Sigma(J_\Sigma - J_\mu \mu^T)$, and the result follows. □

Notice that Theorem 8.15 gives an equivalence, whereas Theorem 8.14 does not. The reason is that the square root of a symmetric positive-definite matrix is not unique. Still, it is canonical; see the discussion in Section 8.4.

As for Theorem 8.14, we can solve Equations (8.28) and (8.29) numerically, and we obtain another algorithm (Algorithm 5 in the Appendix; we will call it GIGO-A), with a behavior similar to the previous one (with Equations (8.26) and (8.27)). For both of them, numerical problems can arise when the covariance matrix is almost singular.

We have not managed to find any example where one of these two algorithms converged to the minimum of the objective function, whereas the other did not, and their behavior is almost the same.

More interestingly, the performances of these two algorithms are also the same as the performances of the exact GIGO algorithm, using the equations of Section 8.3.2.

Notice that even though GIGO-A directly maintains a square root of the covariance matrix, which makes sampling new points easier (to sample a point from $\mathcal{N}(\mu, \Sigma)$, a square root of Σ is needed), both GIGO-Σ and GIGO-A still have to invert the covariance matrix (or its square root) at each step, which is as costly as the decomposition, so one of these algorithms is roughly as expensive to compute as the other.

8.3.2 Explicit Form of the Geodesics of $\mathbb{G}_d$ (from [CO91])

We now give the exact geodesics of $\mathbb{G}_d$: the following results are a rewriting of Theorem 3.1 and its first corollary in [CO91].

Theorem 8.16. *Let $(\dot{\mu}_0, \dot{\Sigma}_0) \in T_{\mathcal{N}(0,I)}\mathbb{G}_d$. The geodesic of $\mathbb{G}_d$ starting from $\mathcal{N}(0,1)$ with initial speed $(\dot{\mu}_0, \dot{\Sigma}_0)$ is given by:*

$$\exp_{\mathcal{N}(0,I)}(s\dot{\mu}_0, s\dot{\Sigma}_0) = \mathcal{N}\left(2R(s)\mathrm{sh}(\frac{sG}{2})G^{-}\dot{\mu}_0, R(s)R(s)^T\right), \tag{8.30}$$

where exp *is the Riemannian exponential of $\mathbb{G}_d$, G is any matrix satisfying:*

$$G^2 = \dot{\Sigma}_0^2 + 2\dot{\mu}_0\dot{\mu}_0^T, \tag{8.31}$$

$$R(s) = \left(\left(\mathrm{ch}(\frac{sG}{2}) - \dot{\Sigma}_0 G^{-}\mathrm{sh}(\frac{sG}{2})\right)^{-1}\right)^T \tag{8.32}$$

and G^{-} is a pseudo-inverse of G

In [CO91], the existence of G (as a square root of $\dot{\Sigma}_0^2 + 2\dot{\mu}_0\dot{\mu}_0^T$) is proven. Notice that, anyway, in the expansions of (8.30) and (8.32), only even powers of G appear.

Additionally, since, for all $A \in GL_d(\mathbb{R})$, for all $\mu_0 \in \mathbb{R}^d$, the application:

$$\begin{array}{rccc} \phi & : & \mathbb{G}_d & \to & \mathbb{G}_d \\ & & \mathcal{N}(\mu, \Sigma) & \mapsto & \mathcal{N}(A\mu + \mu_0, A\Sigma A^T) \end{array} \tag{8.33}$$

preserves the geodesics, we find the general expression for the geodesics of $\mathbb{G}_d$.

Corollary 8.17. *Let $\mu_0 \in \mathbb{R}^d$, $A \in GL_d(\mathbb{R})$ and $(\dot{\mu}_0, \dot{\Sigma}_0) \in T_{\mathcal{N}(\mu_0, A_0A_0^T)}\mathbb{G}_d$. The geodesic of $\mathbb{G}_d$ starting from $\mathcal{N}(\mu, \Sigma)$ with initial speed $(\dot{\mu}_0, \dot{\Sigma}_0)$ is given by:*

$$\exp_{\mathcal{N}(\mu_0, A_0A_0^T)}(s\dot{\mu}_0, s\dot{\Sigma}_0) = \mathcal{N}(\mu_1, A_1A_1^T), \tag{8.34}$$

with:

$$\mu_1 = 2A_0R(s)\mathrm{sh}(\frac{sG}{2})G^{-}A_0^{-1}\dot{\mu}_0 + \mu_0, \tag{8.35}$$

$$A_1 = A_0R(s), \tag{8.36}$$

where exp *is the Riemannian exponential of $\mathbb{G}_d$, G is any matrix satisfying:*

$$G^2 = A_0^{-1}(\dot{\Sigma}_0\Sigma_0^{-1}\dot{\Sigma}_0 + 2\dot{\mu}_0\dot{\mu}_0^T)(A_0^{-1})^T, \tag{8.37}$$

$$R(s) = \left(\left(\mathrm{ch}(\frac{sG}{2}) - A_0^{-1}\dot{\Sigma}_0(A_0^{-1})^T G^{-}\mathrm{sh}(\frac{sG}{2})\right)^{-1}\right)^T, \tag{8.38}$$

and G^{-} is a pseudo-inverse of G.

It should be noted that the final values for mean and covariance do not depend on the choice of G as a square root of:

$$A_0^{-1}(\dot{\Sigma}_0\Sigma_0^{-1}\dot{\Sigma}_0 + 2\dot{\mu}_0\dot{\mu}_0^T)(A_0^{-1})^T.$$

The reason for this is that $\mathrm{ch}(G)$ is a Taylor series in G^2, and so are $\mathrm{sh}(G)G^-$ and $G^-\mathrm{sh}(G)$.

For our practical implementation, we actually used these Taylor series instead of the expression of the corollary.

8.4 Using a Square Root of the Covariance Matrix

The IGO framework (on $\mathbb{G}_d$, for example) emphasizes the Riemannian manifold structure on $\mathbb{G}_d$. All of the algorithms studied here (including GIGO, which is not strictly speaking an IGO algorithm) define a trajectory in $\mathbb{G}_d$ (a new point for each step), and to go from a point θ to the next one (θ'), we follow some curve $\gamma : [0, \delta t] \to \mathbb{G}_d$, with $\gamma(0) = \theta$, $\gamma(\delta t) = \theta'$ and $\dot{\gamma}(0)$ given by the natural gradient ($\dot{\gamma}(0) = \sum_{i=1}^N \hat{w}_i \tilde{\nabla}_\theta P_\theta(x_i) \in T_\theta \mathbb{G}_d$).

To be compatible with this point of view, an algorithm giving an update rule for a square root (any matrix A such that $\Sigma = AA^T$: since we do not force A to be symmetric, the decomposition is not unique) of the covariance matrix A has to satisfy the following condition: for a given initial speed, the covariance matrix $\Sigma^{t+\delta t}$ after one step must depend only on Σ^t and not on the square root A^t chosen for Σ^t.

The xNES algorithm does satisfy this condition: consider two xNES algorithms, with the same learning rates, respectively, at (μ, A_1^t) and (μ, A_2^t), with $A_1^t(A_1^t)^T = A_2^t(A_2^t)^T$ (*i.e.*, they define the same Σ^t), using the same samples x_i to compute the natural gradient update , then we will have $\Sigma_1^{t+\delta t} = \Sigma_2^{t+\delta t}$. Using the definitions of Section 9.2, we have just shown that what we will call the "xNES trajectory" is well defined.

It is also important to notice that, in order to be well defined, a natural gradient algorithm updating a square root of the covariance matrix has to specify more conditions than simply following the natural gradient.

The reason for this is that the natural gradient is a vector tangent to $\mathbb{G}_d$: it lives in a space of dimension $d(d+3)/2$ (the dimension of $\mathbb{G}_d$), whereas the vector (μ, A) lives in a space of dimension $d(d+1)$ (the dimension of $\mathbb{R}^n \times GL_n(\mathbb{R})$), which is too large: there exists infinitely many applications $t \mapsto A_t$, such that a given curve $\gamma : t \mapsto \mathcal{N}(\mu_t, \Sigma_t)$ can be written $\gamma(t) = \mathcal{N}(\mu_t, A_t A_t^T)$. This is why Theorem 8.15 is simply an implication, whereas Theorem 8.14 is an equivalence.

More precisely, let us consider A in $\mathrm{GL}_d(\mathbb{R})$ and v_A, v'_A two infinitesimal updates of A. Since $\Sigma = AA^T$, the infinitesimal update of Σ corresponding to v_A (resp. v'_A) is $v_\Sigma = Av_A^T + v_A A^T$ (resp. $v'_\Sigma = Av'^T_A + v'_A A^T$).

It is now easy to see that v_A and v'_A define the same direction for Σ (*i.e.*, $v_\Sigma = v'_\Sigma$) if and only if $AM^T + MA^T = 0$, where $M = v_A - v'_A$. This is equivalent to $A^{-1}M$ antisymmetric.

For any $A \in \mathrm{M}_d(\mathbb{R})$, let us denote by T_A the space of the matrices M, such that $A^{-1}M$ is antisymmetric or, in other words, $T_A := \{u \in \mathrm{M}_d(\mathbb{R}),\ Au^T + uA^T = 0\}$. Having a subspace S_A in direct sum with T_A for all A is sufficient (but not necessary) to have a well-defined update rule. Namely, consider the (linear) application:

$$\begin{array}{rccc} \phi_A & : & \mathrm{M}_d(\mathbb{R}) & \to & \mathrm{S}_d(\mathbb{R}) \\ & & v_A & \mapsto & Av_A^T + v_A A^T \end{array},$$

sending an infinitesimal update of A to the corresponding update of Σ. It is not bijective, but as we have seen before, $\operatorname{Ker}\phi_A = T_A$, and therefore, if we have, for some U_A,

$$\mathrm{M}_d(\mathbb{R}) = U_A \oplus T_A, \tag{8.39}$$

then $\phi_A|_{U_A}$ is an isomorphism. Let v_Σ be an infinitesimal update of Σ. We choose the following update of A corresponding to v_Σ:

$$v_A := (\phi_A|_{U_A})^{-1}(v_\Sigma). \tag{8.40}$$

Any U_A, such that $U_A \oplus T_A = \mathrm{M}_d(\mathbb{R})$, is a reasonable choice to pick v_A for a given v_Σ. The choice $S_A = \{u \in \mathrm{M}_d(\mathbb{R}),\ Au^T - uA^T = 0\}$ has an interesting additional property; it is the orthogonal of T_A for the norm:

$$\|v_A\|_\Sigma^2 := \operatorname{Tr}(v_A^T \Sigma^{-1} v_A) = \operatorname{Tr}((A^{-1}v_A)^T A^{-1} v_A). \tag{8.41}$$

and consequently, it can be defined without referring to the parametrization, which makes it a canonical choice. To prove this, remark that $T_A = \{M \in \mathrm{M}_d(\mathbb{R}), A^{-1}M \text{ antisymmetric}\}$ and $S_A = \{M \in \mathrm{M}_d(\mathbb{R}), A^{-1}M \text{ symmetric}\}$ and that if M is symmetric and N is antisymmetric, then

$$\operatorname{Tr}(M^T N) = \sum_{i,j=1}^{d} m_{ij} n_{ij} = \sum_{i=1}^{d} m_{ii} n_{ii} + \sum_{1 \leqslant i < j \leqslant d} m_{ij}(n_{ij} + n_{ji}) = 0. \tag{8.42}$$

Let us now show that this is the choice made by xNES and GIGO-A (which are well-defined algorithms updating a square root of the covariance matrix).

Proposition 8.18. *Let $A \in M_n(\mathbb{R})$. The v_A given by the xNES and GIGO-A algorithms lies in $S_A = \{u \in \mathrm{M}_d(\mathbb{R}),\ Au^T - uA^T = 0\} = S_A$.*

Proof. For xNES, let us write $\dot\gamma(0) = (v_\mu, v_\Sigma)$ and $v_A := \frac{1}{2} A G_M$. We have $A^{-1}v_A = \frac{1}{2}G_M$, and therefore, forcing M (and G_M) to be symmetric in xNES is equivalent to $A^{-1}v_A = (A^{-1}v_A)^T$, which can be rewritten as $Av_A^T = v_A A^T$. For GIGO-A, Equation (8.27) shows that $\Sigma_t(J_\Sigma - J_\mu \mu_t^T)$ is symmetric, and with this fact in mind, Equation (8.29) shows that we have $Av_A^T = v_A A^T$ (v_A is $\dot A_t$). □

Chapter 9

Blockwise GIGO, twisted GIGO

9.1 Decoupling the step size

9.1.1 Twisting the Metric

As we can see, the IGO framework does not allow one to recover the learning rates for xNES and pure rank-μ CMA-ES, which is a problem, since usually, the covariance learning rate is set much smaller than the mean learning rate (see either [Han11] or [GSY+10]).

A way to recover these learning rates is to incorporate them directly into the metric (see also blockwise GIGO, in Section 9.1.2). More precisely:

Definition 9.1 (Twisted Fisher metric)**.** *Let $\eta_\mu, \eta_\Sigma \in \mathbb{R}$, and let $(P_\theta)_{\theta\in\Theta}$ be a family of normal probability distributions: $P_\theta = \mathcal{N}(\mu(\theta), \Sigma(\theta))$, with μ and Σ C^1. We call the "(η_μ, η_Σ)-twisted Fisher metric" the metric defined by:*

$$I_{i,j}(\eta_\mu, \eta_\Sigma)(\theta) = \frac{1}{\eta_\mu} \frac{\partial \mu^T}{\partial \theta_i} \Sigma^{-1} \frac{\partial \mu}{\partial \theta_j} + \frac{1}{\eta_\Sigma} \frac{1}{2} \operatorname{tr}\left(\Sigma^{-1} \frac{\partial \Sigma}{\partial \theta_i} \Sigma^{-1} \frac{\partial \Sigma}{\partial \theta_j} \right). \tag{9.1}$$

All of the remainder of this section is simply a rewriting of the work in Chapter 7 with the twisted Fisher metric instead of the regular Fisher metric. We will use the term "twisted geodesic" instead of "geodesic for the twisted metric".

This approach seems to be somewhat arbitrary: arguably, the mean and the covariance play a "different role" in the definition of a Gaussian (only the covariance can affect diversity, for example), but we lack a reasonable intrinsic characterization that would make this choice of twisting more natural. This construction can be slightly generalized (see the Appendix).

The IGO flow and the IGO algorithms can be modified to take into account the twisting of the metric; the (η_μ, η_Σ)-twisted IGO flow reads:

$$\frac{\mathrm{d}\theta^t}{\mathrm{d}t} = I(\eta_\mu, \eta_\Sigma)^{-1}(\theta) \int_X W^f_{\theta^t}(x) \nabla_\theta \ln P_\theta(x) P_{\theta^t}(\mathrm{d}x). \tag{9.2}$$

The only difference with (7.8) is that $I^{-1}(\theta)$ has been replaced by $I(\eta_\mu, \eta_\Sigma)^{-1}(\theta)$.

This leads us to the twisted IGO algorithms.

Definition 9.2. *The (η_μ, η_Σ)-twisted IGO algorithm associated with parametrization θ, sample size N, step size δt and selection scheme w is given by the following update rule:*

$$\theta^{t+\delta t} = \theta^t + \delta t I(\eta_\mu, \eta_\Sigma)^{-1}(\theta^t) \sum_{i=1}^N \hat{w}_i \frac{\partial \ln P_\theta(x_i)}{\partial \theta}.$$

Definition 9.3. *The (η_μ, η_Σ)-twisted geodesic IGO algorithm associated with sample size N, step size δt and selection scheme w is given by the following update rule:*

$$\theta^{t+\delta t} = \exp_{\theta^t}(Y \delta t) \tag{9.3}$$

where:

$$Y = I(\eta_\mu, \eta_\Sigma)^{-1}(\theta^t) \sum_{i=1}^N \hat{w}_i \frac{\partial \ln P_\theta(x_i)}{\partial \theta}. \tag{9.4}$$

By definition, the twisted geodesic IGO algorithm does not depend on the parametrization (but it does depend on η_μ and η_Σ).

There is some redundancy between δt, η_μ and η_Σ: the only values actually appearing in the equations are $\delta t \eta_\mu$ and $\delta t \eta_\Sigma$. More formally:

Proposition 9.4. *Let $k, d, N \in \mathbb{N}$, $\eta_\mu, \eta_\Sigma, \delta t, \lambda_1, \lambda_2 \in \mathbb{R}$ and $w\colon [0;1] \to \mathbb{R}$.*

The (η_μ, η_Σ)-twisted IGO algorithm with sample size N, step size δt and selection scheme w coincides with the $(\lambda_1 \eta_\mu, \lambda_1 \eta_\Sigma)$-twisted IGO algorithm with sample size N, step size $\lambda_2 \delta t$ and selection scheme $\frac{1}{\lambda_1 \lambda_2} w$. The same is true for geodesic IGO.

In order to obtain the twisted algorithms, the Fisher metric in IGO has to be replaced by the metric from Definition 9.1. In practice, the equations found by twisting the metric are exactly the equations without twisting, except that we have "forced" the learning rates η_μ, η_Σ to appear by multiplying the increments of μ and Σ by η_μ and η_Σ.

We can now describe pure rank-μ CMA-ES and xNES with separate learning rates as twisted IGO algorithms:

Proposition 9.5 (xNES as IGO)**.** *The xNES algorithm with sample size N, weights w_i and learning rates $\eta_\mu, \eta_\sigma = \eta_B = \eta_\Sigma$ coincides with the $\frac{\eta_\mu}{\delta t}, \frac{\eta_\Sigma}{\delta t}$-twisted IGO algorithm with sample size N, weights w_i, step size δt and in which, given the current position (μ_t, A_t), the set of Gaussians is parametrized by:*

$$(\delta, M) \mapsto \mathcal{N}\left(\mu_t + A_t\delta, \left(A_t \exp(\frac{1}{2}M)\right)\left(A_t \exp(\frac{1}{2}M)\right)^T\right),$$

with $\delta \in \mathbb{R}^m$ and $M \in \mathrm{Sym}(\mathbb{R}^m)$.

The parameters maintained by the algorithm are (μ, A), and the x_i are sampled from $\mathcal{N}(\mu, AA^T)$.

Proposition 9.6 (Pure rank-μ CMA-ES as IGO)**.** *The pure rank-μ CMA-ES algorithm with sample size N, weights w_i and learning rates η_μ and η_Σ coincides with the $(\frac{\eta_\mu}{\delta t}, \frac{\eta_\Sigma}{\delta t})$-twisted IGO algorithm with sample size N, weights w_i, step size δt and the parametrization (μ, Σ).*

The proofs of these two statements are an easy rewriting of their non-twisted counterparts: one can return to the non-twisted metric (up to a η_Σ factor) by changing μ to $\frac{\sqrt{\eta_\sigma}}{\sqrt{\eta_\mu}}\mu$.

We give the equations of the twisted geodesics of $\mathbb{G}_d$ in the Appendix.

9.1.2 Blockwise GIGO, an almost intrinsic description of xNES

Although xNES is not GIGO, it is possible to define a family of algorithms extending GIGO and including xNES, by decomposing our family of probability distributions as a product and by following the restricted geodesics simultaneously.

Definition 9.7 (Splitting)**.** *Let Θ be a Riemannian manifold. A splitting of Θ is n manifolds $\Theta_1, ..., \Theta_n$ and a diffeomorphism $\Theta \cong \Theta_1 \times ... \times \Theta_n$. If for all $x \in \Theta$, for all $1 \leqslant i < j \leqslant n$, we also have $T_{i,x}M \perp T_{j,x}M$ as subspaces of T_xM (see Notation 9.8), then the splitting is said to be compatible with the Riemannian structure. If the Riemannian manifold is not ambiguous, we will simply write a "compatible splitting".*

We now give some notation, and we define the blockwise GIGO update:

Notation 9.8. *Let Θ be a Riemannian manifold, $\Theta_1, ..., \Theta_n$ a splitting of Θ, $\theta = (\theta_1, ..., \theta_n) \in \Theta$, $Y \in T_\theta\Theta$ and $1 \leqslant i \leqslant n$.*

- *We denote by $\Theta_{\theta,i}$ the Riemannian manifold*

$$\{\theta_1\} \times ... \times \{\theta_{i-1}\} \times \Theta_i \times \{\theta_{i+1}\} \times ... \times \{\theta_n\},$$

with the metric induced from Θ. There is a canonical isomorphism of vector spaces $T_\theta\Theta = \oplus_{i=1}^n T\Theta_{\theta,i}$. Moreover, if the splitting is compatible, it is an isomorphism of Euclidean spaces.

- *We denote by $\Phi_{\theta,i}$ the exponential at θ of the manifold $\Theta_{\theta,i}$.*

Definition 9.9 (Blockwise GIGO update)**.** *Let $\Theta_1, ..., \Theta_n$ be a compatible splitting. The blockwise GIGO algorithm in Θ with splitting $\Theta_1, ..., \Theta_n$ associated with sample size N, step sizes $\delta t_1, ..., \delta t_n$ and selection scheme w is given by the following update rule:*

$$\theta \leftarrow (\theta_1^{t+\delta t_1}, ..., \theta_n^{t+\delta t_n}) \tag{9.5}$$

where:

$$Y = I^{-1}(\theta^t) \sum_{i=1}^{N} \hat{w}_i \frac{\partial \ln P_\theta(x_i)}{\partial \theta}, \tag{9.6}$$

$$\theta_k^{t+\delta t_k} = \Phi_{\theta^t,k}(\delta t_k Y_k), \tag{9.7}$$

with Y_k the $T\Theta_{\theta,k}$-component of Y. This update only depends on the splitting (and not on the parametrization inside each Θ_k).

The compatibility condition ensures that the natural gradient of $W_{\theta^t}^f$ (defined in Section 7.2) in the whole manifold Θ really is the sum of the gradients of this same function in the submanifolds Θ_k. A practical consequence is that the Y_k in Equation (9.7) can be computed simply by taking the natural gradient in Θ_k:

$$Y_k = I_k^{-1}(\theta_i^t) \sum_{i=1}^{N} \hat{w}_i \frac{\partial \ln P_\theta(x_i)}{\partial \theta_k}, \tag{9.8}$$

where I_k is the metric of Θ_k.

Since blockwise GIGO only depends on the splitting (and the tunable parameters: sample size, step sizes and selection scheme), it can be thought of as almost parametrization-invariant.

Notice that blockwise GIGO updates and twisted GIGO updates are two different things: firstly, blockwise GIGO can be defined on any manifold with a compatible splitting, whereas twisted GIGO (and twisted IGO) are only defined for Gaussians. However, even in $\mathbb{G}_d(\eta_\mu, \eta_\Sigma)$, with the splitting (μ, Σ), these two algorithms are different: for instance, if $\eta_\mu = \eta_\Sigma$ and $\delta t = 1$, then the twisted GIGO is the regular GIGO algorithm, whereas blockwise GIGO is not (actually, we will prove that it is the xNES algorithm). The only thing blockwise GIGO and twisted GIGO have in common is that they are compatible with the (η_μ, η_Σ)-twisted IGO flow Equation (9.2): a parameter θ^t following these updates with $\delta t \to 0$ and $N \to \infty$ is a solution of Equation (9.2).

We now have a new description of the xNES algorithm:

Proposition 9.10 (xNES is a Blockwise GIGO algorithm)**.** *The Blockwise GIGO algorithm in $\mathbb{G}_d$ with splitting $\Phi : \mathcal{N}(\mu, \Sigma) \mapsto (\mu, \Sigma)$, sample size N, step sizes $\delta t_\mu, \delta t_\Sigma$ and selection scheme w coincides with the xNES algorithm with sample size N, weights w_i and learning rates $\eta_\mu = \delta t_\mu, \eta_\sigma = \eta_B = \delta t_\Sigma$.*

Proof. Firstly, notice that the splitting (μ, Σ) is compatible, by Proposition 7.2.

Now, let us compute the Blockwise GIGO update: we have $\mathbb{G}_d \cong \mathbb{R}^d \times P_d$, where P_d is the space of real positive-definite matrices of dimension d. We have $\Theta_{\theta^t,1} = (\mathbb{R}^d \times \{\Sigma^t\}) \hookrightarrow \mathbb{G}_d$, $\Theta_{\theta^t,2} = (\{\mu^t\} \times P_d) \hookrightarrow \mathbb{G}_d$. The induced metric on $\Theta_{\theta^t,1}$ is the Euclidean metric, so we have:

$$\mu \leftarrow \mu^t + \delta t_1 Y_\mu.$$

Since we have already shown (using the notation in Definition 7.6) that $Y_\mu = AG_\mu$ (in the proof of Proposition 7.7), we find:

$$\mu \leftarrow \mu^t + \delta t_1 AG_\mu.$$

On $\Theta_{\theta^t,2}$, we have the following Lagrangian for the geodesics:

$$\mathcal{L}(\Sigma, \dot{\Sigma}) = \frac{1}{2} \operatorname{tr}(\dot{\Sigma}\Sigma^{-1}\dot{\Sigma}\Sigma^{-1}).$$

By applying Noether's theorem, we find that

$$J_\Sigma = \Sigma^{-1}\dot{\Sigma}$$

is invariant along the geodesics of $\Theta_{\theta^t,2}$, so they are defined by the equation $\dot{\Sigma} = \Sigma J_\Sigma = \Sigma\Sigma_0^{-1}\dot{\Sigma}_0$ (and therefore, any update preserving the invariant J_Σ will satisfy this first-order differential equation and follow the geodesics of $\Theta_{\theta^t,2}$). The xNES update for the covariance matrix is given by $A(t) = A_0 \exp(tG_M/2)$. Therefore, we have $\Sigma(t) = A_0 \exp(tG_M)A_0^T$, $\Sigma^{-1}(t) = (A_0^{-1})^T \exp(-tG_M)A_0^{-1}$, $\dot{\Sigma}(t) = A_0 \exp(tG_M)G_M A_0^T$ and, finally, $\Sigma^{-1}(t)\dot{\Sigma}(t) = (A_0^{-1})^T G_M A_0^T = \Sigma_0^{-1}\dot{\Sigma}_0$. Therefore, xNES preserves J_Σ, and therefore, xNES follows the geodesics of $\Theta_{\theta^t,2}$ (notice that we had already proven this in Proposition 9.13, since we are looking at the geodesics of $\mathbb{G}_d$ with a fixed mean). □

Although blockwise GIGO is somewhat "less natural" than GIGO, it can be easier to compute for some splittings (as we have just seen), and in the case of the Gaussian distributions, the mean-covariance splitting seems reasonable.

9.2 Trajectories of Different IGO Steps

As we have seen, two different IGO algorithms (or an IGO algorithm and the GIGO algorithm) coincide at first order in δt when $\delta t \to 0$. In this section, we study the differences between pure rank-μ CMA-ES, xNES and GIGO by looking at the second order in δt, and in particular, we show that xNES and GIGO do not coincide in the general case.

We view the updates done by one step of the algorithms as paths on the manifold $\mathbb{G}_d$, from $(\mu(t), \Sigma(t))$ to $(\mu(t+\delta t), \Sigma(t+\delta t))$, where δt is the time step of our algorithms, seen as IGO algorithms. More formally:

Definition 9.11. *1. We call the GIGO update trajectory the application:*

$$T_{\text{GIGO}} \colon (\mu, \Sigma, v_\mu, v_\Sigma) \mapsto \left(\delta t \mapsto \exp_{\mathcal{N}(\mu, AA^T)}(\delta t \eta_\mu v_\mu, \delta t \eta_\Sigma v_\Sigma)\right).$$

*(*exp *is the exponential of the Riemannian manifold $\mathbb{G}_d(\eta_\mu, \eta_\Sigma)$)*

2. We call the xNES update trajectory the application:

$$T_{\text{xNES}} \colon (\mu, \Sigma, v_\mu, v_\Sigma) \mapsto \left(\delta t \mapsto \mathcal{N}(\mu + \delta t \eta_\mu v_\mu, A \exp[\eta_\Sigma \delta t A^{-1} v_\Sigma (A^{-1})^T] A^T)\right),$$

with $AA^T = \Sigma$. The application above does not depend on the choice of a square root A.

3. We call the CMA-ES update trajectory the application:

$$T_{\text{CMA}} \colon (\mu, \Sigma, v_\mu, v_\Sigma) \mapsto \left(\delta t \mapsto \mathcal{N}(\mu + \delta t \eta_\mu v_\mu, AA^T + \delta t \eta_\Sigma v_\Sigma)\right).$$

These applications map the set of tangent vectors to $\mathbb{G}_d$ ($T\mathbb{G}_d$) to the curves in $\mathbb{G}_d(\eta_\mu, \eta_\Sigma)$.

We will also use the following notation: $\mu_{\text{GIGO}} := \phi_\mu \circ T_{GIGO}$, $\mu_{\text{xNES}} := \phi_\mu \circ T_{xNES}$, $\mu_{\text{CMA}} := \phi_\mu \circ T_{CMA}$, $\Sigma_{\text{GIGO}} := \phi_\Sigma \circ T_{GIGO}$, $\Sigma_{\text{xNES}} := \phi_\Sigma \circ T_{xNES}$ and $\Sigma_{\text{CMA}} := \phi_\Sigma \circ T_{CMA}$, where ϕ_μ (resp. ϕ_Σ) extracts the μ-component (resp. the Σ-component) of a curve.

In particular, $\text{Im}(\phi_\mu) \subset \mathbb{R}^d$ and $\text{Im}(\phi_\Sigma) \subset P_d$, where P_d (the set of real symmetric positive-definite matrices of dimension d) is seen as a subset of $\mathbb{R}^{d^2}$.

For instance, $T_{\text{GIGO}}(\mu, \Sigma, v_\mu, v_\Sigma)(\delta t)$ gives the position (mean and covariance matrix) of the GIGO algorithm after a step of size δt, while μ_{GIGO} and Σ_{GIGO} give, respectively, the mean component and the covariance component of this position.

This formulation ensures that the trajectories we are comparing had the same initial position and the same initial speed, which is the case provided the sampled points (the values directly sampled from $\mathcal{N}(\mu, \Sigma)$, not from $\mathcal{N}(0, I)$ and transformed) are the same.

Different IGO algorithms coincide at first order in δt. The following proposition gives the second order expansion of the trajectories of the algorithms.

Proposition 9.12 (Second derivatives of the trajectories)**.** *We have:*

$$\mu_{\text{GIGO}}(\mu, \Sigma, v_\mu, v_\Sigma)''(0) = \eta_\mu \eta_\Sigma v_\Sigma \Sigma_0^{-1} v_\mu,$$

$$\mu_{\text{xNES}}(\mu, \Sigma, v_\mu, v_\Sigma)''(0) = \mu_{\text{CMA}}(\mu, \Sigma, v_\mu, v_\Sigma)''(0) = 0,$$

$$\Sigma_{\text{GIGO}}(\mu, \Sigma, v_\mu, v_\Sigma)''(0) = \eta_\Sigma^2 v_\Sigma \Sigma^{-1} v_\Sigma - \eta_\mu \eta_\Sigma v_\mu v_\mu^T,$$

$$\Sigma_{\text{xNES}}(\mu, \Sigma, v_\mu, v_\Sigma)''(0) = \eta_\Sigma^2 v_\Sigma \Sigma^{-1} v_\Sigma,$$

$$\Sigma_{\text{CMA}}(\mu, \Sigma, v_\mu, v_\Sigma)''(0) = 0.$$

Proof. We can immediately see that the second derivatives of μ_{xNES}, μ_{CMA} and Σ_{CMA} are zero. Next, we have:

$$\begin{aligned}
\Sigma_{\text{xNES}}(\mu, \Sigma, v_\mu, v_\Sigma)(t) &= A \exp[tA^{-1}\eta_\Sigma v_\Sigma (A^{-1})^T] A^T \\
&= AA^T + t\eta_\Sigma v_\Sigma + \frac{t^2}{2}\eta_\Sigma^2 v_\Sigma (A^{-1})^T A^{-1} v_\Sigma + o(t^2) \\
&= \Sigma + t\eta_\Sigma v_\Sigma + \frac{t^2}{2}\eta_\Sigma^2 v_\Sigma \Sigma^{-1} v_\Sigma + o(t^2).
\end{aligned}$$

The expression of $\Sigma_{\text{xNES}}(\mu, \Sigma, v_\mu, v_\Sigma)''(0)$ follows.

Now, for GIGO, let us consider the geodesic starting at (μ_0, Σ_0) with initial speed $(\eta_\mu v_\mu, \eta_\Sigma v_\Sigma)$. By writing $J_\mu(0) = J_\mu(t)$, we find $\dot{\mu}(t) = \Sigma(t)\Sigma_0^{-1}\dot{\mu}_0$. We then easily have $\ddot{\mu}(0) = \dot{\Sigma}_0\Sigma_0^{-1}\dot{\mu}_0$. In other words:

$$\mu_{\text{GIGO}}(\mu, \Sigma, v_\mu, v_\Sigma)''(0) = \eta_\mu \eta_\Sigma v_\Sigma \Sigma_0^{-1} v_\mu.$$

Finally, by using Theorem 8.14 and differentiating, we find:

$$\ddot{\Sigma} = \eta_\Sigma \dot{\Sigma}(J_\Sigma - J_\mu \mu^T) - \eta_\Sigma \Sigma J_\mu \dot{\mu}^T,$$

$$\ddot{\Sigma}_0 = \eta_\Sigma \dot{\Sigma}_0 \frac{1}{\eta_\Sigma}\Sigma_0^{-1}\dot{\Sigma}_0 - \frac{\eta_\Sigma}{\eta_\mu}\dot{\mu}_0\dot{\mu}_0^T = \eta_\Sigma^2 v_\Sigma \Sigma_0^{-1} v_\Sigma - \eta_\Sigma \eta_\mu v_\mu v_\mu^T.$$

□

In order to interpret these results, we will look at what happens in dimension one. In higher dimensions, we can suppose that the algorithms exhibit a similar behavior, but an exact interpretation is more difficult for GIGO in $\mathbb{G}_d$.

- In [GSY+10], it has been noted that xNES converges to quadratic minima slower than CMA-ES and that it is less subject to premature convergence. That fact can be explained by observing that the mean update is exactly the same for CMA-ES and xNES, whereas xNES tends to have a higher variance (Proposition 9.12 shows this at order two, and it is easy to see that in dimension one, for any μ, Σ, v_μ, v_Σ, we have $\Sigma_{\text{xNES}}(\mu, \Sigma, v_\mu, v_\Sigma) > \Sigma_{\text{CMA}}(\mu, \Sigma, v_\mu, v_\Sigma)$).

- At order two, GIGO moves the mean faster than xNES and CMA-ES if the standard deviation is increasing and more slowly if it is decreasing. This seems to be a reasonable behavior (if the covariance is decreasing, then the algorithm is presumably close to a minimum, and it should not leave the area too quickly). This remark holds only for isolated steps, because we do not take into account the evolution of the variance.
- The geodesics of $\mathbb{G}_1$ are half-circles (see Figure 9.1 below; we recall that $\mathbb{G}_1$ is the Poincaré half-plane). Consequently, if the mean is supposed to move (which always happens), then $\sigma \to 0$ when $\delta t \to \infty$. For example, a step whose initial speed has no component on the standard deviation will always decrease it. See also Proposition 10.1, about the optimization of a linear function.
- For the same reason, for a given initial speed, the update of μ always stays bounded as a function of δt: it is not possible to make one step of the GIGO algorithm go further than a fixed point by increasing δt. Still, the geodesic followed by GIGO changes at each step, so the mean of the overall algorithm is not bounded.

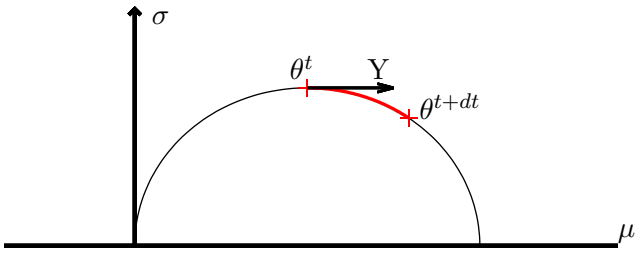


Figure 9.1: One step of the geodesic IGO (GIGO) update.

We now show that xNES follows the geodesics of $\mathbb{G}_d$ if the mean is fixed, but that xNES and GIGO do not coincide otherwise.

Proposition 9.13 (xNES is not GIGO in the general case)**.** *Let $\mu, v_\mu \in \mathbb{R}^d$, $A \in \mathrm{GL}_d$, $v_\Sigma \in \mathrm{M}_d$.*

Then, the GIGO and xNES updates starting at $\mathcal{N}(\mu, \Sigma)$ with initial speeds v_μ and v_Σ follow the same trajectory if and only if the mean remains constant. In other words:

$T_{\mathrm{GIGO}}(\mu, \Sigma, v_\mu, v_\Sigma) = T_{\mathrm{xNES}}(\mu, \Sigma, v_\mu, v_\Sigma)$ *if and only if* $v_\mu = 0$.

Proof. If $v_\mu = 0$, then we can compute the GIGO update by using Theorem 8.14: since $J_\mu = 0$, $\dot{\mu} = 0$, and μ remains constant. Now, we have $J_\Sigma = \Sigma^{-1}\dot{\Sigma}$; this is enough information to compute the update. Since this quantity is also preserved by the xNES algorithm (see, for example, the proof of Proposition 9.10), the two updates coincide.

If $v_\mu \neq 0$, then $\Sigma_{\text{xNES}}(\mu, \Sigma, v_\mu, v_\Sigma)''(0) - \Sigma_{\text{GIGO}}(\mu, \Sigma, v_\mu, v_\Sigma)''(0) = \eta_\mu \eta_\Sigma v_\mu v_\mu^T \neq 0$ and, in particular, $T_{\text{GIGO}}(\mu, \Sigma, v_\mu, v_\Sigma) \neq T_{\text{xNES}}(\mu, \Sigma, v_\mu, v_\Sigma)$. □

Chapter 10

Numerical experiments

We conclude this part with some numerical experiments to compare the behavior of GIGO, xNES and pure rank-μ CMA-ES (we give the pseudocodes for these algorithms in the Appendix). We made two series of tests. The first one is a performance test, using classical benchmark functions and the settings from [GSY+10]. The goal of the second series of tests is to illustrate the computations in Section 9.2 by plotting the trajectories (standard deviation *versus* mean) of these three algorithms in dimension one.

The source code is available at `https://www.lri.fr/~bensadon`.

10.1 Benchmarking

For the first series of experiments, presented in Figure 10.1, we used the following parameters, taken from [GSY+10] (we recall that xNES and pure rank-μ CMA-ES are seen as IGO algorithms):

- Varying dimension.
- Sample size: $\lfloor 4 + 3\log(d) \rfloor$.
- Weights: $w_i = \frac{\max(0, \log(\frac{n}{2}+1) - \log(i))}{\sum_{j=1}^{N} \max(0, \log(\frac{n}{2}+1) - \log(j))} - \frac{1}{N}$.
- IGO step size and learning rates: $\delta t = 1, \eta_\mu = 1, \eta_\Sigma = \frac{3}{5}\frac{3+\log(d)}{d\sqrt{d}}$..
- Initial position: $\theta^0 = \mathcal{N}(x_0, I)$, where x_0 is a random point of the circle with center zero, and radius 10.
- Euler method for GIGO: Number of steps: 100. We used the GIGO-A variant of the algorithm. No significant difference was noticed with GIGO-Σ or with the exact GIGO algorithm. The only advantage of having an explicit solution of the geodesic equations is that the update is quicker to compute.
- We chose not to use the exact expression of the geodesics for this benchmarking to show that having to use the Euler method is fine. However, we did run the tests, and the results are basically the same as GIGO-A.

We plot the median number of runs to achieve target fitness (10^{-8}). Each algorithm has been tested in dimension 2, 4, 8, 16, 32 and 64: a missing point means that all runs converged prematurely.

10.1.1 Failed Runs

In Figure 10.1, a point is plotted even if only one run was successful. Below is the list of the settings for which at least one run converged prematurely.

- Only one run reached the optimum for the cigar-tablet function with CMA-ES in dimension eight.
- Seven runs (out of 24) reached the optimum for the Rosenbrock function with CMA-ES in dimension 16.
- About half of the runs reached the optimum for the sphere function with CMA-ES in dimension four.

For the following settings, all runs converged prematurely.

- GIGO did not find the optimum of the Rosenbrock function in any dimension.
- CMA-ES did not find the optimum of the Rosenbrock function in dimension 2, 4, 32 and 64.
- All of the runs converged prematurely for the cigar-tablet function in dimension two with CMA-ES, for the sphere function in dimension two for all algorithms and for the Rosenbrock function in dimension two and four for all algorithms.

Dimension	d	From 2 to 64
Sample size	N	$4 + 3\log(d)$
Weights	$(w_i)_{i\in[1,N]}$	$\frac{\max(0,\log(\frac{n}{2}+1)-\log(i)}{\sum_{j=1}^{N}\max(0,\log(\frac{n}{2}+1)-\log(j)} - \frac{1}{N}$
IGO step size	δt	1
Mean learning rate	η_μ	1
Covariance learning rate	η_Σ	$\frac{3}{5}\frac{3+\log(d)}{d\sqrt{d}}$
Euler step-size (for GIGO only)	h	0.01(100 steps)
GIGO implementation		GIGO-A
Sphere function		$x \mapsto \sum_{i=1}^{d} x_i^2$
Cigar-tablet		$x \mapsto x_1^2 + \sum_{i=2}^{d-1} 10^4 x_i^2 + 10^8 x_d^2$
Rosenbrock		$x \mapsto \sum_{i=1}^{d-1}(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$
x-axis		Dimension
y-axis		Number of function calls to reach fitness 10^{-8}.

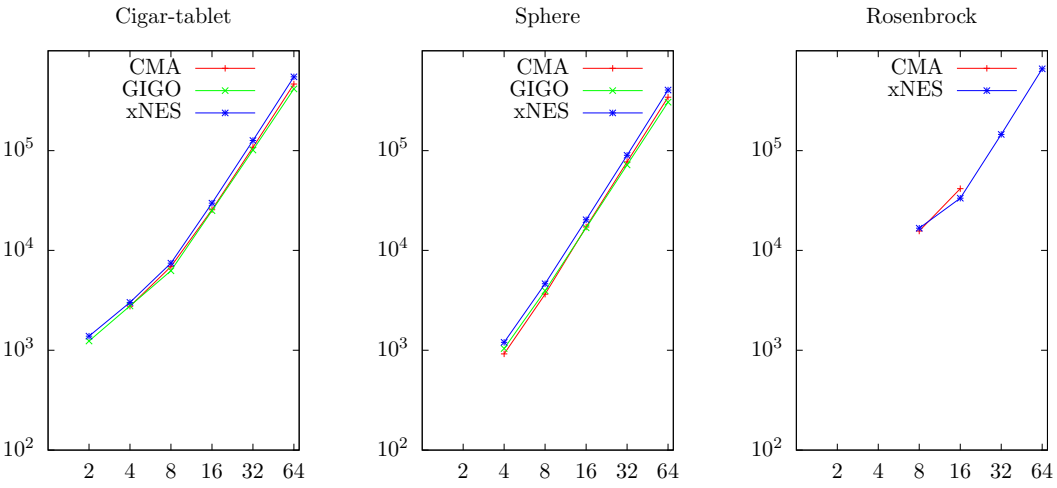


Figure 10.1: Median number of function calls to reach 10^{-8} fitness on 24 runs for: sphere function, cigar-tablet function and Rosenbrock function. Initial position $\theta^0 = \mathcal{N}(x_0, I)$, with x_0 uniformly distributed on the circle of center zero and radius 10. We recall that the "CMA-ES" algorithm here is using the so-called pure rank-μ CMA-ES update.

10.1.2 Discussion

As the last item in Section 10.1.1 shows, all of the algorithms converge prematurely in a low dimension, probably because the covariance learning

rate has been set too high (or because the sample size is too small). This is different from the results in [GSY+10].

This remark aside, as noted in [GSY+10], the xNES algorithm shows more robustness than CMA-ES and GIGO: it is the only algorithm able to find the minimum of the Rosenbrock function in high dimensions. However, its convergence is consistently slower.

In terms of performance, when both of them work, pure rank-μ CMA-ES (or equivalently, IGO in the parametrization (μ, Σ)) and GIGO are extremely close (GIGO is usually a bit better). An advantage of GIGO is that it is theoretically defined for any δt, η_Σ, whereas the covariance matrix maintained by CMA-ES (not only pure rank-μ CMA-ES) can stop being positive definite if $\eta_\Sigma \delta t > 1$. However, in that case, the GIGO algorithm is prone to premature convergence (remember Figure 9.1 and see Proposition 10.1 below), and in practice, the learning rates are much smaller.

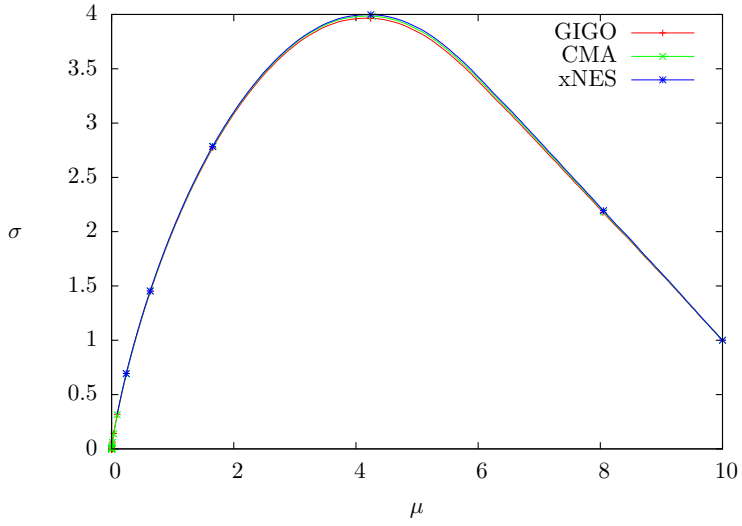
10.2 Plotting Trajectories in $\mathbb{G}_1$

We want the second series of experiments to illustrate the remarks about the trajectories of the algorithms in Section 9.2, so we decided to take a large sample size to limit randomness, and we chose a fixed starting point for the same reason. We use the weights below because of the property of quantile improvement proven in [AO13]: the 1/4-quantile will improve at each step. The parameters we used were the following:

- Sample size: $\lambda = 5,000$
- Dimension one only.
- Weights: $w = 4\mathbf{1}_{q \leqslant 1/4}$ $(w_i = 4.\mathbf{1}_{i \leqslant 1,250})$
- IGO step size and learning rates: $\eta_\mu = 1, \eta_\Sigma = \frac{3}{5}\frac{3+\log(d)}{d\sqrt{d}} = 1.8$, varying δt.
- Initial position: $\theta^0 = \mathcal{N}(10, 1)$
- Dots are placed at $t = 0,\ 1,\ 2\ldots$ (except for the graph $\delta t = 1.5$, for which there is a dot for each step).

Figures 10.2–10.6 show the optimization of $x \mapsto x^2$, and Figures 10.7–10.9 show the optimization of $x \mapsto -x$.

Figure 10.2: Trajectories of GIGO, CMA and xNES optimizing $x \mapsto x^2$ in dimension one with $\delta t = 0.01$, sample size 5000, weights $w_i = 4.\mathbf{1}_{i \leqslant 1250}$ and learning rates $\eta_\mu = 1$, $\eta_\Sigma = 1.8$. One dot every 100 steps. All algorithms exhibit a similar behavior



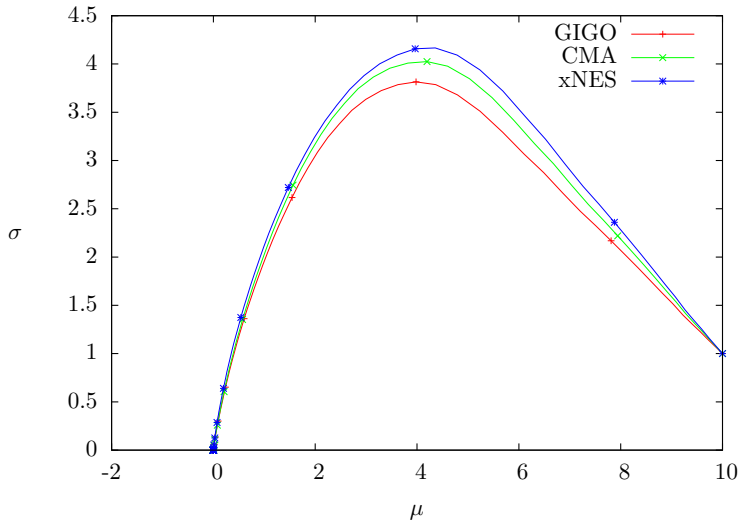


Figure 10.4: Trajectories of GIGO, CMA and xNES optimizing $x \mapsto x^2$ in dimension one with $\delta t = 0.1$, sample size 5000, weights $w_i = 4.\mathbf{1}_{i \leqslant 1250}$ and learning rates $\eta_\mu = 1$, $\eta_\Sigma = 1.8$. One dot every 10 steps. All algorithms exhibit a similar behavior, and differences start to appear. It cannot be seen on the graph, but the algorithm closest to zero after 400 steps is CMA ($\sim 1.10^{-16}$, followed by xNES ($\sim 6.10^{-16}$) and GIGO ($\sim 2.10^{-15}$).

Figure 10.3: Trajectories of GIGO, CMA and xNES optimizing $x \mapsto x^2$ in dimension one with $\delta t = 0.5$, sample size 5000, weights $w_i = 4.\mathbf{1}_{i \leqslant 1250}$ and learning rates $\eta_\mu = 1$, $\eta_\Sigma = 1.8$. One dot every two steps. Stronger differences. Notice that after one step, the lowest mean is still GIGO (~ 8.5, whereas xNES is around 8.75), but from the second step, GIGO has the highest mean, because of the lower variance.

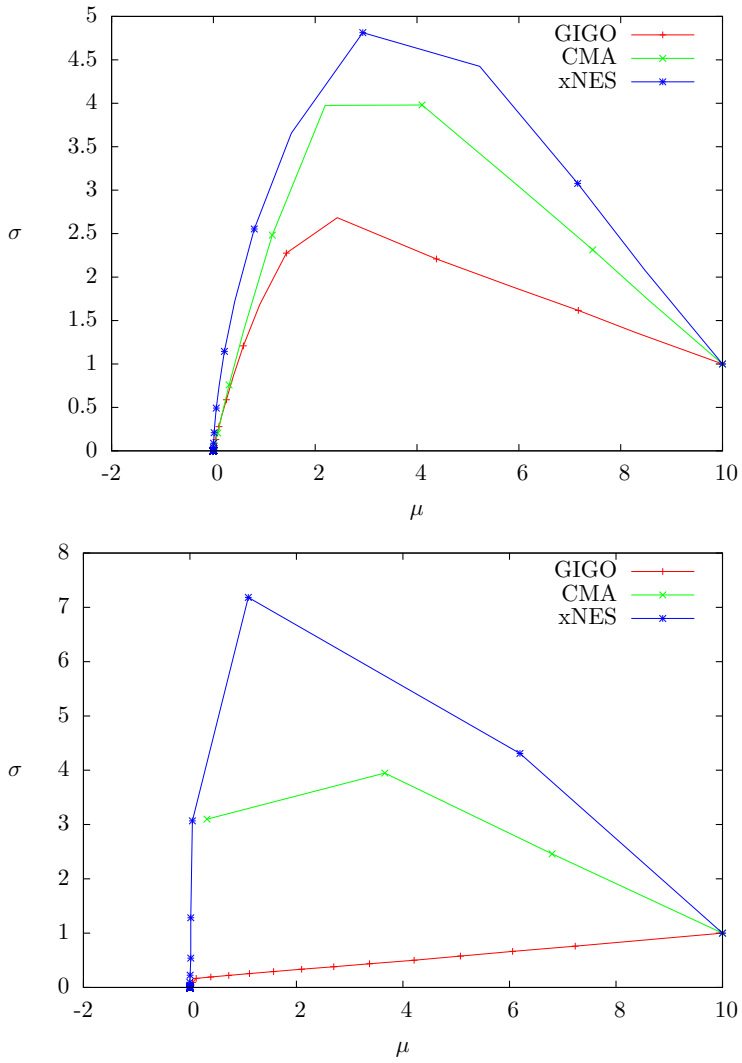
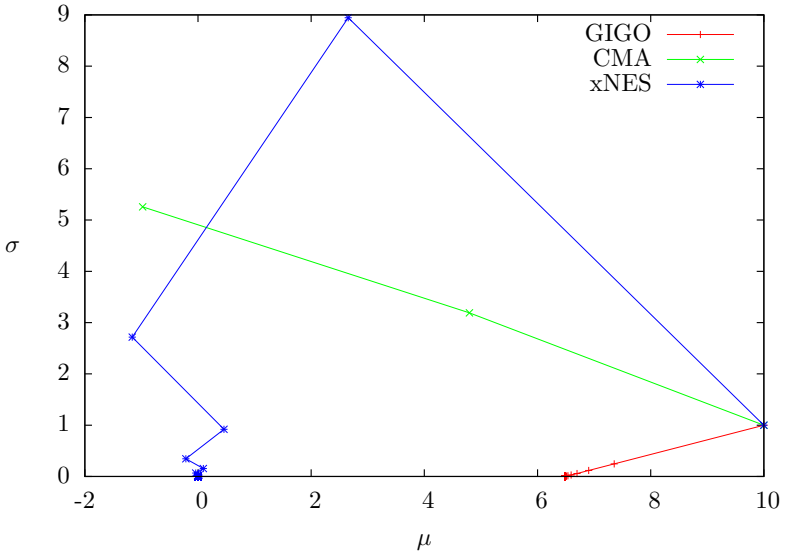


Figure 10.5: Trajectories of GIGO, CMA and xNES optimizing $x \mapsto x^2$ in dimension one with $\delta t = 1$, sample size 5000, weights $w_i = 4.\mathbf{1}_{i \leqslant 1250}$ and learning rates $\eta_\mu = 1$, $\eta_\Sigma = 1.8$. One dot per step. The CMA-ES algorithm fails here, because at the fourth step, the covariance matrix is not positive definite anymore (it is easy to see that the CMA-ES update is always defined if $\delta t \eta_\Sigma < 1$, but this is not the case here). Furthermore, notice (see also Proposition 10.1) that at the first step, GIGO decreases the variance, whereas the σ-component of the IGO speed is positive.

Figure 10.6: Trajectories of GIGO, CMA and xNES optimizing $x \mapsto x^2$ in dimension one with $\delta t = 1.5$, sample size 5000, weights $w_i = 4.\mathbf{1}_{i \leqslant 1250}$ and learning rates $\eta_\mu = 1$, $\eta_\Sigma = 1.8$. One dot per step. Same as $\delta t = 1$ for CMA. GIGO converges prematurely.



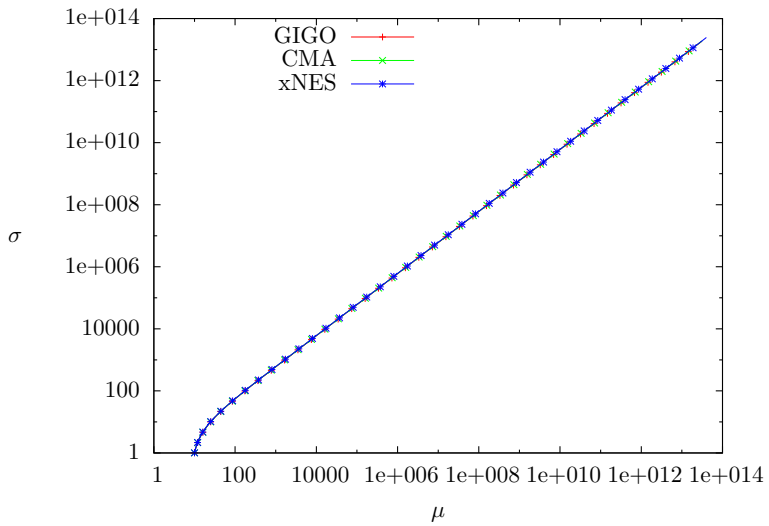
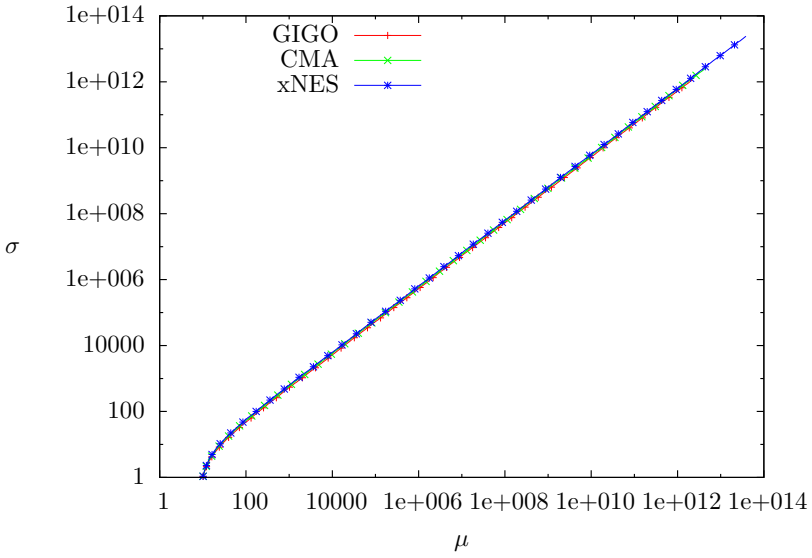


Figure 10.7: Trajectories of GIGO, CMA and xNES optimizing $x \mapsto -x$ in dimension one with $\delta t = 0.01$, sample size 5000, weights $w_i = 4.\mathbf{1}_{i \leqslant 1250}$ and learning rates $\eta_\mu = 1$, $\eta_\Sigma = 1.8$. One dot every 100 steps. Almost the same for all algorithms.

Figure 10.8: Trajectories of GIGO, CMA and xNES optimizing $x \mapsto -x$ in dimension one with $\delta t = 0.1$, sample size 5000, weights $w_i = 4.\mathbf{1}_{i \leqslant 1250}$ and learning rates $\eta_\mu = 1$, $\eta_\Sigma = 1.8$. One dot every 10 steps. It is not obvious on the graph, but xNES is faster than CMA, which is faster than GIGO.



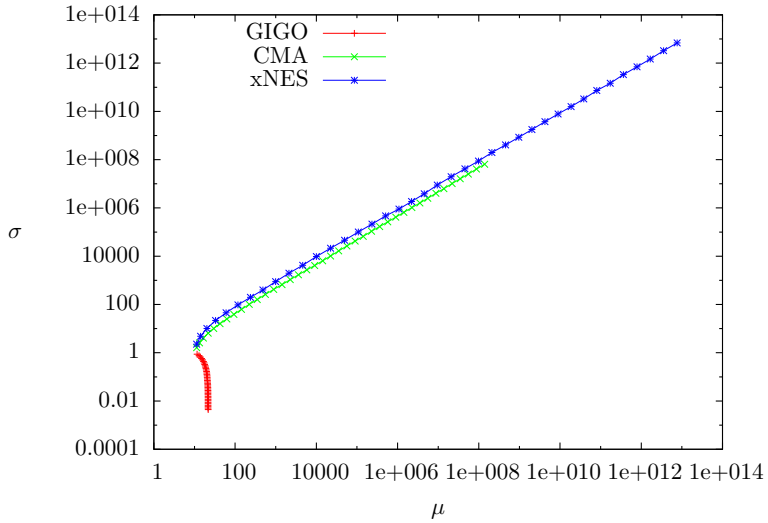


Figure 10.9: Trajectories of GIGO, CMA and xNES optimizing $x \mapsto -x$ in dimension one with $\delta t = 1$, sample size $5,000$, weights $w_i = 4.\mathbf{1}_{i \leqslant 1,250}$ and learning rates $\eta_\mu = 1$, $\eta_\Sigma = 1.8$. One dot per step. GIGO converges, for the reasons discussed earlier.

Figures 10.5, 10.6 and 10.9 show that when $\delta t \geqslant 1$, GIGO reduces the covariance, even at the first step. More generally, when using the GIGO algorithm in $\tilde{\mathbb{G}}_d$ for the optimization of a linear function, there exists a critical step size δt_{cr} (depending on the learning rates η_μ, η_σ and on the weights w_i), above which, GIGO will converge, and we can compute its value when the weights are of the form $\mathbf{1}_{q \leqslant q_0}$ (for $q_0 \geqslant 0.5$, the discussion is not relevant, because in that case, even the IGO flow converges prematurely. Compare with the critical δt of the smoothed cross entropy method and IGO-ML in [OAAH11]).

Proposition 10.1. *Let $d \in \mathbb{N}$, k, η_μ, $\eta_\sigma \in \mathbb{R}_+^*$; let $w = k.\mathbf{1}_{q \leqslant q_0}$; and let*

$$\begin{array}{rccc} g & : & \mathbb{R}^d & \to & \mathbb{R} \\ & & x & \mapsto & -x_1 \end{array}.$$

Let μ_n be the first coordinate of the mean, and let σ_n^2 be the variance (at step n) maintained by the (η_μ, η_σ)-twisted geodesic IGO algorithm in $\tilde{\mathbb{G}}_d$ associated with selection scheme w, sample size ∞ [1] and step size δt, when optimizing g.

There exists δt_{cr}, such that:

- *if $\delta t > \delta t_{cr}$, (σ_n) converges to zero with exponential speed and (μ_n) converges.*
- *if $\delta t = \delta t_{cr}$, (σ_n) remains constant and (μ_n) tends to ∞ with linear speed.*
- *if $0 < \delta t < \delta t_{cr}$, both (σ_n) and μ_n tend to ∞ with exponential speed.*

The proof and the expression of δt_{cr} can be found in the Appendix.

In the case corresponding to $k = 4$, $n = 1$, $q_0 = 1/4$, $\eta_\mu = 1$, $\eta_\sigma = 1.8$, we find:

$$\delta t_{\text{cr}} \approx 0.84. \tag{10.1}$$

[1] It has been proven in [OAAH11] that IGO algorithms are consistent with the IGO flow, *i.e.*, they get closer and closer to the IGO flow as sample size tends to infinity. In other words:

$$\lim_{N \to \infty} \sum_{i=1}^{N} \hat{w}_i \frac{\partial \ln P_\theta(x_i)}{\partial \theta} = \tilde{\nabla}_\theta \int_X W^f_{\theta^t(x)} P_\theta(\mathrm{d}x).$$

Sample size ∞ means that we replace the IGO speed in the GIGO update by its limit for large N. In particular, it is deterministic.

Conclusion

1 Summary

In this thesis, we introduced two different algorithms, generic Context Tree Weighting and GIGO, motivated respectively by minimum description length considerations and invariance principles.

1.1 Expert Trees

By defining a new formal context, we generalized the Context Tree Weighting and Context Tree Switching algorithms, obtained an elegant proof of the property of the former, and introduced edgewise versions of these algorithms. As for the original CTW, the new algorithms introduced here consider simultaneously a large quantity of models explaining the data, and, for the generalized CTW, computes Bayesian posteiors efficiently.

We also investigated the behavior of context tree switching algorithms, using different switch distributions than the ones used in [VNHB11], and showed that for regression, the corresponding Context Tree Switching algorithms achieve better performance than Context Tree Weighting or Bayesian histograms on a specific sequence.

1.2 GIGO

We introduced GIGO, a fully parametrization-invariant black-box optimization algorithm, and, thanks to Noether's theorem, we obtained first order equations for the geodesics of $\mathbb{G}_d$.

Moreover, we described xNES as a "blockwise GIGO" algorithm, giving it an almost intrinsic description.

2 Future directions

The following questions are still open:

GIGO on other manifolds. We only studied GIGO on $\mathbb{G}_d$. Geodesics are usually hard to compute, except in some cases, such as the manifold of Bernoulli distributions. However, in that case the length of the geodesics

is finite, so definition problems arise. Noether's theorem could be useful to obtain equations for computing GIGO updates on other manifolds.

Stochastic IGO. (mentioned in [OAAH11]) The IGO algorithm can be seen as a stochastic process $\theta^{t+\delta t} = \theta^t + \delta t X_N^t$, where X_N^t is computed from N random samples of $f(x_i)$. Because of computation time considerations, it is natural to take $N \propto \delta t$ (we want the same number of calls of the objective function f to go from θ^t to θ^{t+T}).

The stochastic differential equation obtained when $\delta t \to 0$, with $N \propto \delta t$ should be easier to study theoretically than the actual IGO update.

From experts to specialists. In order to recover the specialists framework, it is necessary to define a Bayesian combination for experts with domains that are not necessarily identical. It cannot be done directly since by definition, experts are not penalized when data falls outside of their domain, so it seems necessary to define a "domain extension" for experts.

Parameters of the switch distribution in expert trees. Our numerical tests were conducted with generic settings for the switch distributions in the expert trees. Putting more prior mass on sequences starting closed and opening at some time (i.e. of the form $0^n 1^\infty$, $n \in [0, \infty)$) might yield better results, since this seems to be the "reasonable behaviour". This remark can be extended to switch distributions out of expert trees: more prior mass should be put to sequences that go from simple models to more complex (in the sense that they have more parameters to estimate) ones. Experiments are needed to confirm or infirm this intuition.

Testing expert tree algorithms on real data, with more refined experts. The expert tree algorithms introduced in part II are a general framework for expert aggregation, and as such, they could be used in a large variety of situations. It would be interesting to study their performance with more sophisticated experts at the leaves.

Appendix A

Expert Trees

Lemma A.1 (Useful computation)**.** *Let $K \in \mathbb{N}$. We have:*

$$\sum_{k=0}^{K-1} k2^k = K2^K + O(2^K). \tag{A.1}$$

Proof.

$$\begin{aligned}
\sum_{k=0}^{K-1} k2^k &= (K-1)\sum_{k=0}^{K-1} 2^k - \sum_{k=0}^{K-2} 2^k - ... - \sum_{k=0}^{0} 2^k && \text{(A.2)}\\
&= (K-1)(2^K - 1) - (2^{K-1} - 1) - ... - (2^1 - 1) && \text{(A.3)}\\
&= K2^K - 2^K - K + 1 - (2^{K-1} - 1) - ... - (2^1 - 1) && \text{(A.4)}\\
&= K2^K + O(2^K), && \text{(A.5)}
\end{aligned}$$

□

Corollary A.2. *Let $K \in \mathbb{N}$, let $t \in (2^{K-1}, 2^K]$.*
We have:

$$\ln 2 \sum_{k=0}^{K-2} k2^k + (t - 2^{K-1})(K-1)\ln 2 = t\ln t + O(t) \tag{A.6}$$

Proof. We have, by Lemma A.1 (since $O(t) = O(2^K) = O(2^{K-1})$):

$$\ln 2 \sum_{k=0}^{K-2} k2^k + (t - 2^{K-1})(K-1)\ln 2 = 2^{K-1}(K-1)\ln 2 + (t - 2^{K-1})(K-1)\ln 2 + O(t) \tag{A.7}$$

$$\begin{aligned}
&= t(K-1)\ln 2 + O(t) && \text{(A.8)}\\
&= t\ln t + O(t), && \text{(A.9)}
\end{aligned}$$

which is what we wanted. □

Lemma A.3. *Let X, Y be two sets, Let $\mathbf{T}$ be an infinite binary edgewise weighted expert tree from X to Y such that for all $s \in \mathbf{T}$, the domain of $\mathcal{E}_s$ is fixed*[1]*, and $\mathrm{Tar}(\mathcal{E}_s) = Y$. Let $\mathbf{x} = (x_1, x_2, ...) \in X^{\mathbb{N}}$, let $(y_1, y_2, ...) \in Y^{\mathbb{N}}$, let $z_k = (x_k, y_k)$ for all $k \in \mathbb{N}^*$, and let $\mathcal{I} = \mathcal{I}(\mathbf{x})$ be an edgewise switching pattern such that for any $s \in \mathbf{T}$, and $t \in c(s)$*

- $\mathcal{I}_{s\to t} = 0^\infty$ *if* $|s| = K - 1$.
- $\mathcal{I}_{s\to t} = 01^\infty$ *otherwise.*

Then, to predict y_n, $\mathcal{E}_\mathcal{I}$ tries to use the deepest expert that has seen at least one other data point before x_n.

In other words, for all $n > 1$, $\mathcal{E}_\mathcal{I}(z_n|z^{n-1}) = \mathcal{E}_{s(x^n)}(z_n|z^{n-1})$, where $s(x^n) \in \mathbf{T}$ is defined by: $x_n \in \mathrm{Dom}(\mathcal{E}_{s(x^n)})$, $x^{n-1} \cap \mathrm{Dom}(\mathcal{E}_{s(x^n)}) \neq \varnothing$, and either $|s(x^n)| = K - 1$ or $x^{n-1} \cap \mathrm{Dom}(\mathcal{E}_{t(x^n)}) = \varnothing$, where $t(x^n)$ is the child of $s(x^n)$ containing x_n.

Proof. Firstly, we recall that by Definition 4.40:

$\mathcal{E}_{\mathcal{I}_{s\to t}}(y|\mathbf{z}, x) := \mathbf{1}_{(\mathcal{I}_{s\to t})_{\phi_t(\mathbf{z})}=0}(\mathcal{E}_s)_{|\mathrm{Dom}(\mathcal{E}_t)}(y|\mathbf{z}, x) + \mathbf{1}_{(\mathcal{I}_{s\to t})_{\phi_t(\mathbf{z})}=1}\mathcal{E}_{\mathcal{I},t}(y|\mathbf{z}, x)$, where $\phi_t(z^n) = |\{x_k | 1 \leqslant k \leqslant n, x_k \in \mathrm{Dom}(\mathcal{E}_t)\}|$, if we use the fixed domain convention, and the fact that $\mathrm{Dom}(\mathcal{E}_t) = Y$.

Let $n \in \mathbb{N}$ and let us write $s = s(x^n)$.

Now, let $s' \in \mathbf{T}$ such that $|s'| < |s|$ (in particular, $|s'| < K$), and $x_n \in \mathrm{Dom}(\mathcal{E}_{s'})$ (i.e. s is a descendent of s'); and let t' be the child of s' such that $x_n \in \mathrm{Dom}(\mathcal{E}_{t'})$.

By definition of $s(x^n)$, $x^{n-1} \cap \mathrm{Dom}(\mathcal{E}_{s(x^n)}) \neq \varnothing$. Consequently, x_n is at least the second point in $\mathrm{Dom}(\mathcal{E}_{t'})$. Since $\mathcal{I}_{s'\to t'} = 01^\infty$, $(\mathcal{I}_{s'\to t'})_{1+|x^{n-1}\cap \mathrm{Dom}(\mathcal{E}_{t'})|} = 1$, so $\mathcal{E}_{\mathcal{I},s'}(x_n|x^{n-1}) = \mathcal{E}_{\mathcal{I},t'}(x_n|x^{n-1})$.

We have just show that for any ascendent s' of s, we have $\mathcal{E}_{\mathcal{I},s'}(x_n|x^{n-1}) = \mathcal{E}_{\mathcal{I},t'}(x_n|x^{n-1})$, where t' is the child of s' which is also an ascendent of s (or s itself). Consequently, $\mathcal{E}_\mathcal{I} = \mathcal{E}_{\mathcal{I},\varepsilon} = \mathcal{E}_{\mathcal{I},s(x^n)}$.

Moreover, either $x^{n-1} \cap \mathrm{Dom}(\mathcal{E}_t(x^n)) = \varnothing$ or $|s(x^n)| = K - 1$. In both cases, $(\mathcal{I}_{s(x_n)\to t(x_n)})_{1+|x^{n-1}\cap \mathrm{Dom}(\mathcal{E}_{t(x^n)})|} = 0$, so $\mathcal{E}_{\mathcal{I},s(x_n)}(x_n|x^{n-1}) = \mathcal{E}_{s(x_n)}(x_n|x^{n-1})$, which concludes the proof.

□

A.1 Balanced sequences

Proposition A.4. *Let $K > 1$. If (x_i) is well K-balanced, and I, J are two intervals such that $|I| = |J|$, then the first point in I appears before the $K^2 + 1$-st point in J. In other words, if t_I is the index of the first point in I:*

$$|\{i | 1 \leqslant i < t_I, x_i \in J\}| \leqslant K^2 \tag{A.10}$$

[1] So formally, $\mathrm{Dom}(\mathcal{E}_s) = (X \times Y)^* \times D_s$ for some D_s, but we will write $\mathrm{Dom}(\mathcal{E}_s) = D_s$.

Proof. Let $N_J(t)$ be the number of points in J at time t. We prove that $N_J(t) > K^2$ implies $N_I(t) \geqslant 1$.

By definition, $N_J(t) \leqslant K|J|t$, so $N_J(t) > K^2$ implies $K^2 < K|J|t = K|I|t$ (since $N_J(t) > 1$), so $1 < \frac{1}{K}|I|t$ and therefore $1 \leqslant \lfloor \frac{1}{K}|I|t \rfloor \leqslant N_I(t)$. □

A.1.1 Specific sequence achieveing the bound in Section 5.3.1

We recall that $(u_n)_{n\in\mathbb{N}}$ is the sequence defined by: $u_0 = 0$, and for all k, m, if $0 \leqslant m < 2^k$, $u_{2^k+m} = \frac{2m+1}{2^{k+1}}$.

If $0 \leqslant m < 2^k$, we write $I_{k,m} := [\frac{m}{2^k}, \frac{m+1}{2^k})$

Lemma A.5. *Let $k, k' \in \mathbb{N}$, $0 \leqslant m < 2^k$, $0 \leqslant m' < 2^{k'}$.*
$u_{2^k+m} \in I_{k',m'}$ iff $m' = \lfloor 2^{k'-k-1}(2m+1) \rfloor$.

Proof.

$$u_{2^k+m} \in I_{k',m'} \Leftrightarrow m' = \max\{n \in \mathbb{N}, \frac{n}{2^{k'}} \leqslant \frac{2m+1}{2^{k+1}}\} \tag{A.11}$$

$$\Leftrightarrow m' = \max\{n \in \mathbb{N}, n \leqslant 2^{k'-k-1}(2m+1)\}, \tag{A.12}$$

which is what we wanted. □

and

Lemma A.6. *We have, for all $k > 0$, $0 \leqslant m < 2^k$:*

1. *u_{2^k+m} is the second point in $I_{k,m}$: $|\{k', m' | 2^{k'} + m' < 2^k + m, u_{2^{k'}+m'} \in I_{k,m}\}| = 1$*
2. *u_{2^k+m} is at least the third point in $I_{k-1,\lfloor m/2 \rfloor}$: $|\{k', m' | 2^{k'} + m' < 2^k + m, u_{2^{k'}+m'} \in I_{k-1,\lfloor m/2 \rfloor}\}| \geqslant 2$*

Proof. Firstly notice that for all k for all $0 \leqslant m < k$, $I_{k,m} = I_{k+1,2m} \cup I_{k+1,2m+1}$. Consequently, if $k > 0$, $I_{k,m} \subset I_{k-1,\lfloor m/2 \rfloor}$.

2 is therefore a consequence of 1 applied to $I_{k-1,\lfloor m/2 \rfloor}$ (the second point in $I_{k-1,\lfloor m/2 \rfloor}$ was $u_{2^{k-1}+\lfloor m/2 \rfloor}$, so u_{2^k+m} is at least the third).

Now, we can see that $\{u_0\} \cup \{u_{2^{k'}+m'} | k' < k, 0 \leqslant m' < 2^{k'}\} = \{\frac{n}{2^k} | 0 \leqslant n < 2^k\}$ (by induction: The odd n are by definition the $u_{2^{k-1}+m}$, and the even n come frome the previous $u_{2^{k'}+m'}$). Since $I_{k,m}$ is exactly of length $\frac{1}{2^k}$, there is exactly one of the $u_{2^{k'}+m'}$, $k' < k$, $0 \leqslant m' < 2^{k'}$ in $I_{k,m}$.

Finally, for $m' < m$, $u_{2^k+m'} = \frac{2m'+1}{2^{k+1}} < \frac{m}{2^k}$, so $u_{2^k+m'} \notin I_{k,m}$, which concludes the proof. □

A.2 Loss of Normal–Gamma experts

This section contains various bounds on the loss incurred by Normal–Gamma experts. They are used in section 5.3.

Lemma A.7 (General loss for a Normal–Gamma expert)**.** *Let $d \in \mathbb{N}$, $D \in \mathbb{R}^d$, λ, α, $\beta \in \mathbb{R}$, $D \subset \mathbb{R}^d$, and let $\mathcal{E}_{D,\mu_0,\lambda,\alpha,\beta}$ be a Normal-Gamma expert on D with regularization μ_0, λ, α, β. We have:*

$$-\ln \mathcal{E}_{D,\mu_0,\lambda,\alpha,\beta}(\mathbf{z}|\varnothing) = (\alpha+t/2)\ln(\frac{t+\lambda}{2}) - \ln\Gamma(\alpha+t/2) + (\alpha+t/2)\ln V_0 + \frac{1}{2}\ln(t+\lambda) - \frac{1}{2}\ln\lambda + \frac{t}{2}\ln(2\pi) + \ln\Gamma(\alpha) - \alpha\ln\beta, \tag{A.13}$$

where $V_0 = \frac{2\beta+\sum y_i^2}{t+\lambda} - (\frac{\sum y_i}{t+\lambda})^2$.

Proof.

$$\mathcal{E}_{D,0,\lambda,\alpha,\beta}(\mathbf{z}|\varnothing) = \int_\mu \int_\tau \frac{\beta^\alpha\sqrt{\lambda}}{\Gamma(\alpha)\sqrt{2\pi}} \tau^{\alpha-\frac{1}{2}} e^{-\beta\tau} e^{-\frac{\lambda\tau(\mu-\mu_0)^2}{2}} P_{\mu,\tau}(\mathbf{y}) \mathrm{d}\tau \mathrm{d}\mu \tag{A.14}$$

$$= \frac{\beta^\alpha\sqrt{\lambda}}{\Gamma(\alpha)\sqrt{2\pi}^{t+1}} \int_\tau \tau^{\alpha+t/2-1/2} e^{-\beta\tau} \int_\mu e^{-\frac{\lambda\tau\mu^2}{2}} e^{-\frac{\tau}{2}\sum(y_i-\mu)^2} \mathrm{d}\mu \mathrm{d}\tau \tag{A.15}$$

$$:= \frac{\beta^\alpha\sqrt{\lambda}}{\Gamma(\alpha)\sqrt{2\pi}^{t+1}} \int_\tau \tau^{\alpha+t/2-1/2} e^{-\beta\tau} I_\tau \mathrm{d}\tau, \tag{A.16}$$

where $I_\tau = \int_\mu e^{-\frac{\lambda\tau\mu^2}{2}} e^{-\frac{\tau}{2}\sum(y_i-\mu)^2} \mathrm{d}\mu$. Now,

$$I_\tau = \int_\mu e^{-\frac{\lambda\tau\mu^2}{2}} e^{-\frac{\tau}{2}\sum(y_i-\mu)^2} \mathrm{d}\mu = e^{-\frac{\tau(t+\lambda)}{2}(\frac{\sum y_i^2}{t+\lambda} - (\frac{\sum y_i}{t+\lambda})^2)} \sqrt{\frac{2\pi}{(t+\lambda)\tau}}, \tag{A.17}$$

so we have:

$$\mathcal{E}_{D,0,\lambda,\alpha,\beta}(\mathbf{z}|\varnothing) = \frac{\beta^\alpha\sqrt{\lambda}}{\Gamma(\alpha)\sqrt{2\pi}^{t}\sqrt{(t+\lambda)}} \int_\tau \tau^{\alpha+t/2-1} e^{-\beta\tau} e^{-\frac{\tau(t+\lambda)}{2}(\frac{\sum y_i^2}{t+\lambda} - (\frac{\sum y_i}{t+\lambda})^2)} \mathrm{d}\tau \tag{A.18}$$

$$= \frac{\beta^\alpha\sqrt{\lambda}}{\Gamma(\alpha)\sqrt{2\pi}^{t}\sqrt{(t+\lambda)}} \int_\tau \tau^{\alpha+t/2-1} e^{-\left(\beta + \frac{t+\lambda}{2}(\frac{\sum y_i^2}{t+\lambda} - (\frac{\sum y_i}{t+\lambda})^2)\right)\tau} \mathrm{d}\tau \tag{A.19}$$

$$= \frac{\beta^\alpha\sqrt{\lambda}}{\Gamma(\alpha)\sqrt{2\pi}^{t}\sqrt{(t+\lambda)}} \int_\tau \tau^{\alpha+t/2-1} e^{-\frac{t+\lambda}{2}V_0\tau} \mathrm{d}\tau \tag{A.20}$$

$$= \frac{\beta^\alpha\sqrt{\lambda}}{\Gamma(\alpha)\sqrt{2\pi}^{t}\sqrt{(t+\lambda)}} \frac{\Gamma(\alpha+t/2)}{(\frac{t+\lambda}{2})^{\alpha+t/2} V_0^{\alpha+t/2}} \tag{A.21}$$

The lemma follows. □

We also add a useful decomposition of V_0:

Lemma A.8 (Another form of V_0)**.** *Let $t \in \mathbb{N}$, $\beta, \lambda > 0$, $(y_i) \in \mathbb{R}^t$. We have:*

$$V_0 := \frac{2\beta + \sum y_i^2}{t+\lambda} - \left(\frac{\sum y_i}{t+\lambda}\right)^2 = \frac{2\beta}{t+\lambda} + \frac{\lambda}{t+\lambda}\frac{\sum y_i^2}{t+\lambda} + \left(\frac{t}{t+\lambda}\right)^2 \mathrm{Var}(y_i) \tag{A.22}$$

Proof.

$$\frac{2\beta + \sum y_i^2}{t+\lambda} - \left(\frac{\sum y_i}{t+\lambda}\right)^2 = \frac{2\beta}{t+\lambda} + \left(\frac{\sum y_i^2}{t+\lambda} - \frac{t^2}{(t+\lambda)^2}\frac{\sum y_i^2}{t}\right) + \frac{t^2}{(t+\lambda)^2}\frac{\sum y_i^2}{t} - \left(\frac{t}{t+\lambda}\right)^2 \left(\frac{\sum y_i}{t}\right)^2 \tag{A.23}$$

$$= \frac{2\beta}{t+\lambda} + \frac{\lambda}{t+\lambda}\frac{\sum y_i^2}{t+\lambda} + \left(\frac{t}{t+\lambda}\right)^2 \mathrm{Var}(y_i) \tag{A.24}$$

□

We can extract the important terms under Condition 5.8 in the loss of the Normal-Gamma experts computed in Lemma A.7:

Lemma A.9 (Loss of Normal–Gamma experts satisfying Condition 5.8)**.** *Let $t \in \mathbb{R}$, let $\mathcal{S}$ be a set of Normal-Gamma experts satisfying Condition 5.8 such that λ is bounded on $\mathcal{S}$. We have, for all $z^t = (x, y)^t$ such that $x^t \in \mathrm{Dom}(\mathcal{E})^t$:*

$$-\ln \mathcal{E}(z^t|\varnothing) = (\alpha + t/2)\ln V_0 + \frac{1+\ln 2\pi}{2}t + \ln t - \frac{1}{2}\ln\lambda - \alpha\ln\beta + O(1), \tag{A.25}$$

uniformly on $\mathcal{S}$, where λ, α, β are the parameters of $\mathcal{E}$, and $V_0 = \frac{2\beta+\sum y_i^2}{t+\lambda} - (\frac{\sum y_i}{t+\lambda})^2$.

Proof. By Lemma A.7:

$$-\ln \mathcal{E}(z^t|\varnothing) = (\alpha+t/2)\ln(\frac{t+\lambda}{2}) - \ln\Gamma(\alpha+t/2) + (\alpha+t/2)\ln V_0 + \frac{1}{2}\ln(t+\lambda) - \frac{1}{2}\ln\lambda + \frac{t}{2}\ln(2\pi) + \ln\Gamma(\alpha) - \alpha\ln\beta. \tag{A.26}$$

By Stirling's formula:

$$(\alpha + t/2)\ln(\frac{t+\lambda}{2}) - \ln\Gamma(\alpha + t/2) = (\alpha + t/2)\ln(\frac{t+\lambda}{2}) - (\alpha + \frac{t-1}{2})\ln(\alpha + t/2) + (\alpha + t/2) + O(1) \tag{A.27}$$

$$= (\alpha + t/2)(\ln(\frac{t+\lambda}{2}) - \ln(\frac{t+2\alpha}{2})) + t/2 + \frac{1}{2}\ln(\alpha + t/2) + O(1) \tag{A.28}$$

$$= (\alpha + t/2)((1 + \frac{\lambda}{t}) - (1 + \frac{2\alpha}{t}) + o(1/t)) + t/2 + \frac{1}{2}\ln(\alpha + t/2) + O(1) \tag{A.29}$$

$$= t/2 + \frac{1}{2}\ln t + O(1). \tag{A.30}$$

Since α is a constant, we can conclude. □

Lemma A.10 (Uniform upper bound on the empirical average cost of a point under condition 5.8)**.** *Let $t \in \mathbb{R}$, let $\mathcal{S}$ be a set of Normal-Gamma experts satisfying Condition 5.8 with $\lambda(\mathcal{E}) = |\mathrm{Dom}(\mathcal{E})|^2$, such that λ is bounded on $\mathcal{S}$. We have, for all $f : I \to \mathbb{R}$ K-Lipschitz, for all $z^t = (x, y)^t$ such that $x^t \in \mathrm{Dom}(\mathcal{E})^t$:*

$$-\frac{1}{t-1}\ln \mathcal{E}(z_2, ..., z_t|z_1) \leqslant \ln|\mathrm{Dom}(\mathcal{E})| + O(1), \tag{A.31}$$

uniformly on $\mathcal{S}$, where $z_n := (x_n, f(x_n))$.

Proof. Let $\mathcal{E} \in \mathcal{S}$, and let us write $\varepsilon = |\mathrm{Tar}(\mathcal{E})|$. By Lemma A.7, if we write $V_0 = \frac{2\beta + \sum f(x_i)^2}{t+\lambda} - (\frac{\sum f(x_i)}{t+\lambda})^2$:

$$-\ln \mathcal{E}(z_2, ..., z_t|z_1) = (\alpha + t/2)\ln(\frac{t+\lambda}{2}) - \ln\Gamma(\alpha + t/2) + (\alpha + t/2)\ln V_0(t) \tag{A.32}$$

$$-(\alpha + 1/2)\ln(\frac{1+\lambda}{2}) + \ln\Gamma(\alpha + 1/2) - (\alpha + 1/2)\ln V_0(1) \tag{A.33}$$

$$+\frac{1}{2}\ln(t+\lambda) - \frac{1}{2}\ln\lambda + \frac{t}{2}\ln(2\pi) + \ln\Gamma(\alpha) - \alpha\ln\beta \tag{A.34}$$

$$-\frac{1}{2}\ln(1+\lambda) + \frac{1}{2}\ln\lambda - \frac{1}{2}\ln(2\pi) - \ln\Gamma(\alpha) + \alpha\ln\beta \tag{A.35}$$

$$= d_1 + d_0 + \frac{1}{2}\ln(\frac{t+\lambda}{1+\lambda}) + \frac{t-1}{2}\ln(2\pi) \tag{A.36}$$

$$= d_0 + d_1 + O(t), \tag{A.37}$$

where $d_1 = (\alpha + t/2)\ln(\frac{t+\lambda}{2}) - \ln\Gamma(\alpha + t/2) - (\alpha + 1/2)\ln(\frac{1+\lambda}{2}) + \ln\Gamma(\alpha + 1/2)$, and $d_0 = (\alpha + t/2)\ln V_0(t) - (\alpha + 1/2)\ln V_0(1)$. We have, by Stirling formula,

$$d_1 = (\alpha + t/2)\ln(\frac{t+\lambda}{2}) - (\alpha + 1/2)\ln(\frac{1+\lambda}{2}) - \ln\Gamma(\alpha + t/2) + \ln\Gamma(\alpha + 1/2) \tag{A.38}$$

$$= (\alpha + t/2)\ln(\frac{t+\lambda}{1+\lambda}) - (\alpha + (t-1)/2)\ln(\alpha + t/2) + O(t) \tag{A.39}$$

$$= \frac{t}{2}\left(\ln(\frac{t+\lambda}{1+\lambda}) - \ln(\alpha + t/2)\right) + O(t) = O(t). \tag{A.40}$$

and

$$d_0 = (\alpha + t/2)\ln V_0(t) - (\alpha + 1/2)\ln V_0(1) \tag{A.41}$$

$$= (\alpha + t/2)\ln V_0(t) - (\alpha + 1/2)\ln V_0(t) + (\alpha + \frac{1}{2})\ln V_0(t) - (\alpha + 1/2)\ln V_0(1) \tag{A.42}$$

$$= \frac{t-1}{2}\ln V_0(t) + (\alpha + \frac{1}{2})(\ln V_0(t) - \ln V_0(1)) \tag{A.43}$$

Now, since $|I| = \varepsilon$, and f is K-Lipschitz, $\mathrm{Var}(f(x_i)) \leqslant (K\frac{\varepsilon}{2})^2$, and $\lambda = \varepsilon^2$, so:

$$V_0(t) = \frac{2\beta}{t+\lambda} + \lambda\frac{1}{t+\lambda}\frac{\sum f(x_i)^2}{t+\lambda} + (\frac{t}{t+\lambda})^2\mathrm{Var}(f(x_i)) \tag{A.44}$$

$$= \varepsilon^2\frac{2\beta}{\lambda(t+\lambda)} + \varepsilon^2\frac{1}{t+\lambda}\frac{\sum f(x_i)^2}{t+\lambda} + (\frac{t}{t+\lambda})^2\mathrm{Var}(f(x_i)) \tag{A.45}$$

$$\leqslant \varepsilon^2(\frac{\beta}{\lambda}\frac{2}{(t+\lambda)} + \frac{1}{t+\lambda}\max_I(f)^2 + (\frac{t}{t+\lambda})^2\frac{K^2}{4}). \tag{A.46}$$

Consequently $\ln V_0(t) \leqslant 2\ln\varepsilon + \mathrm{cst}$.

Moreover

$$V_0(1) = \frac{2\beta}{1+\lambda} + \lambda\frac{1}{1+\lambda}\frac{f(x_0)^2}{1+\lambda} \tag{A.47}$$

$$\geqslant \varepsilon^2(\frac{2B}{(1+\lambda)^2}). \tag{A.48}$$

Therefore, we have:

$$\ln V_0(t) - \ln V_0(1) \leqslant O(1) \tag{A.49}$$

Consequently, we have $d_1 \leqslant O(t)$ and $d_0 \leqslant (t-1)\ln\varepsilon + O(t)$, and therefore,

$$-\ln\mathcal{E}(z_2, ..., z_t|z_1) \leqslant (t-1)\ln\varepsilon + O(t), \tag{A.50}$$

which is what we wanted. □

More generally, we have:

Lemma A.11 (Uniform upper bound on the empirical average cost of a point under condition 5.8, with an arbitrary number of previous points)**.** *Let $t, u \in \mathbb{R}$, let $\mathcal{S}$ be a set of Normal-Gamma experts satisfying Condition 5.8 with $\lambda(\mathcal{E}) = |\mathrm{Dom}(\mathcal{E})|^2$, such that λ is bounded on $\mathcal{S}$. We have, for all $f : I \to \mathbb{R}$ K-Lipschitz, for all $z^t \in (\mathrm{Dom}(\mathcal{E}) \times \mathbb{R})^t$, for all $(z')^u \in (\mathrm{Dom}(\mathcal{E}) \times \mathbb{R})^u$ with $u \geqslant 1$:*

$$-\frac{1}{t}\ln\mathcal{E}(z^t|(z')^u) \leqslant \ln|\mathrm{Dom}(\mathcal{E})| + O(1) + O(\frac{1}{t}u\ln u), \tag{A.51}$$

uniformly on $\mathcal{S}$, where $z_n := (x_n, f(x_n))$ and $z'_n := (x'_n, f(x'_n))$.

Proof. Let $\mathcal{E} \in \mathcal{S}$, and let us write $\varepsilon = |\mathrm{Dom}(\mathcal{E})|$. By Lemma A.7, if we write $V_0 = \frac{2\beta+\sum f(x_i)^2}{t+\lambda} - (\frac{\sum f(x_i)}{t+\lambda})^2$:

$$-\ln \mathcal{E}(z^t|(z')^u) = (\alpha + (t+u)/2)\ln(\frac{t+u+\lambda}{2}) - \ln\Gamma(\alpha+(t+u)/2) + (\alpha+(t+u)/2)\ln V_0(t+u) \tag{A.52}$$

$$- (\alpha+u/2)\ln(\frac{u+\lambda}{2}) + \ln\Gamma(\alpha+u/2) - (\alpha+u/2)\ln V_0(u) \tag{A.53}$$

$$+ \frac{1}{2}\ln(t+u+\lambda) - \frac{1}{2}\ln\lambda + \frac{t+u}{2}\ln(2\pi) + \ln\Gamma(\alpha) - \alpha\ln\beta \tag{A.54}$$

$$- \frac{1}{2}\ln(u+\lambda) + \frac{1}{2}\ln\lambda - \frac{u}{2}\ln(2\pi) - \ln\Gamma(\alpha) + \alpha\ln\beta \tag{A.55}$$

$$= d_1 + d_0 + \frac{1}{2}\ln(\frac{t+u+\lambda}{u+\lambda}) + \frac{t-1}{2}\ln(2\pi) \tag{A.56}$$

$$= d_0 + d_1 + O(t), \tag{A.57}$$

where $d_1 = (\alpha+(t+u)/2)\ln(\frac{t+\lambda}{2}) - \ln\Gamma(\alpha+(t+u)/2) - (\alpha+u/2)\ln(\frac{u+\lambda}{2}) + \ln\Gamma(\alpha+u/2)$, and $d_0 = (\alpha+(t+u)/2)\ln V_0(t+u) - (\alpha+u/2)\ln V_0(u)$. We have, by Stirling formula,

$$d_1 = (\alpha + (t+u)/2)\ln(\frac{t+u+\lambda}{2}) - (\alpha+u/2)\ln(\frac{u+\lambda}{2}) - \ln\Gamma(\alpha+(t+u)/2) + \ln\Gamma(\alpha+u/2) \tag{A.58}$$

$$= (\alpha+(t+u)/2)\ln\left(\frac{t+u+\lambda}{2\alpha+t+u}\right) - (\alpha+u/2)\ln\left(\frac{u+\lambda}{2\alpha+u}\right) + O(t) \tag{A.59}$$

$$= \frac{u}{2}(\ln\left(\frac{t+u+\lambda}{2\alpha+t+u}\right) - \ln\left(\frac{u+\lambda}{2\alpha+u}\right)) + O(t) \tag{A.60}$$

$$= \frac{u}{2}(\ln\left(1+\frac{\lambda-2\alpha}{2\alpha+t+u}\right) - \ln\left(1+\frac{\lambda-2\alpha}{2\alpha+u}\right)) + O(t) \tag{A.61}$$

$$= \frac{u}{2}\left(\frac{\lambda-2\alpha}{2\alpha+t+u} - \frac{\lambda-2\alpha}{2\alpha+u} + O(\frac{1}{u^2})\right) + O(t) \tag{A.62}$$

$$= O(t). \tag{A.63}$$

and

$$d_0 = (\alpha+(t+u)/2)\ln V_0(t+u) - (\alpha+u/2)\ln V_0(u) \tag{A.64}$$

$$= (\alpha+(t+u)/2)\ln V_0(t+u) - (\alpha+u/2)\ln V_0(t+u) + (\alpha+\frac{u}{2})\ln V_0(t+u) - (\alpha+u/2)\ln V_0(u) \tag{A.65}$$

$$= \frac{t}{2}\ln V_0(t+u) + (\alpha+\frac{u}{2})(\ln V_0(t+u) - \ln V_0(u)) \tag{A.66}$$

Now, since $|I| = \varepsilon$, and f is K-Lipschitz, $\mathrm{Var}(f(x_i)) \leqslant (K\frac{\varepsilon}{2})^2$ for any (x_i), and $\lambda = \varepsilon^2$, so for any T:

$$V_0(T) = \frac{2\beta}{T+\lambda} + \lambda\frac{1}{T+\lambda}\frac{\sum f(x_i)^2}{T+\lambda} + (\frac{T}{T+\lambda})^2\mathrm{Var}(f(x_i)) \tag{A.67}$$

$$= \varepsilon^2\frac{2\beta}{\lambda(T+\lambda)} + \varepsilon^2\frac{1}{T+\lambda}\frac{\sum f(x_i)^2}{T+\lambda} + (\frac{T}{T+\lambda})^2\mathrm{Var}(f(x_i)) \tag{A.68}$$

$$\leqslant \varepsilon^2(\frac{\beta}{\lambda}\frac{2}{(T+\lambda)} + \frac{1}{T+\lambda}\max_I(f)^2 + (\frac{T}{T+\lambda})^2\frac{K^2}{4}). \tag{A.69}$$

Consequently $\ln V_0(T) \leqslant 2\ln\varepsilon + \mathrm{cst}$ for any T.

Moreover

$$V_0(u) = \frac{2\beta}{u+\lambda} + \lambda\frac{1}{u+\lambda}\frac{\sum_{i=1}^u f(x_i)^2}{u+\lambda} + (\frac{u}{u+\lambda})^2\mathrm{Var}_{1\leqslant i\leqslant u}(f(x_i)) \tag{A.70}$$

$$\geqslant \frac{2\beta}{u+\lambda} \tag{A.71}$$

$$\geqslant \frac{2B\lambda}{(u+\lambda)(1+\lambda)} \tag{A.72}$$

$$\geqslant \varepsilon^2\frac{1}{u+\lambda}\frac{2B}{1+\lambda}, \tag{A.73}$$

so $\ln V_0(u) \geqslant 2\ln\varepsilon - \ln(u) + \mathrm{cst}$ and finally

$$\ln V_0(t+u) - \ln V_0(u) \leqslant \ln(u) + O(1) \tag{A.74}$$

Consequently, we have $d_1 \leqslant O(t)$ and $d_0 \leqslant t\ln\varepsilon + O(t) + O(u\ln u)$, and therefore,

$$-\ln\mathcal{E}(z^t|(z')^u) \leqslant t\ln\varepsilon + O(t) + O(u\ln u), \tag{A.75}$$

which is what we wanted. □

Lemma A.12 (Uniform lower bound on the average loss at interval size ε with n observations under condition 5.8)**.** *Let $\mathcal{S}$ be a set of Normal-Gamma experts satisfying Condition 5.8 such that $\lambda(\mathcal{E}) = |\mathrm{Dom}(\mathcal{E})|^2$ and $\mathcal{E} \mapsto |\mathrm{Dom}(\mathcal{E})|$ is bounded on $\mathcal{S}$. We have, for any $\mathcal{E} \in \mathcal{S}$, for any $(z^n) \in (\mathrm{Dom}(\mathcal{E}) \times \mathbb{R})^n$*

$$\bar{L}_{n,\varepsilon} := \frac{1}{n}\mathcal{E}(z^n|\varnothing) \geqslant (1-\frac{1}{n})\ln|\mathrm{Dom}(\mathcal{E})| - \frac{1}{2}\ln n + O(1), \tag{A.76}$$

where $O(1)$ is uniform on $\mathcal{S}$.

Proof. Let $\mathcal{E} \in \mathcal{S}$, and let $\varepsilon := |\mathrm{Dom}(\mathcal{E})|$. By Lemma A.9, we have:

$$-\ln\mathcal{E}(z^n|\varnothing) = (\alpha+n/2)\ln V_0 + \frac{1+\ln 2\pi}{2}n + \ln n - \frac{1}{2}\ln\lambda - \alpha\ln\beta + O(1), \tag{A.77}$$

where, if $z_k = (x_k, y_k)$, $V_0 = \frac{2\beta + \sum y_i^2}{n+\lambda} - (\frac{\sum y_i}{n+\lambda})^2 \geqslant \frac{2\beta}{n+\lambda}$, and by Condition 5.8, we have: $\alpha = \text{cst}$, $\beta = B\varepsilon^2 + o(\varepsilon^2)$, so $\ln \beta = 2 \ln \varepsilon + O(1)$.

Consequently:

$$-\frac{1}{n} \ln \mathcal{E}(z^n | \varnothing) \geqslant (\frac{\alpha}{n} + 1/2) \ln \frac{2\beta}{n+\lambda} - \frac{1}{2n} \ln \lambda - \frac{\alpha}{n} \ln \beta + O(1) \quad \text{(A.78)}$$

$$\geqslant (\frac{\alpha}{n} + 1/2)(2 \ln \varepsilon - \ln n) - \frac{1}{n} \ln \varepsilon - 2\frac{\alpha}{n} \ln \varepsilon + O(1), \quad \text{(A.79)}$$

which is what we wanted. □

A.3 Pseudocode for CTW

Notation for the algorithm below:

The latest data point is x_n, the previous data is $z^{n-1} = (x, y)^{n-1}$, and we want to predict $f(x_n)$.

Each node s of the tree contains, in addition to the expert $\mathcal{E}_s$: two weights $w_s \in [0, 1]$ the posterior weights for closing the node s,$n_s \in \mathbb{N}$ (initialized to the prior weight), the number of data points that have already fallen in $\text{Dom}(\mathcal{E}_s)$, and $x_s \in s$ (useful only if $n_s = 1$, the exact value of the data point in $\text{Dom}(\mathcal{E}_s)$.

Algorithm 2 Infinite depth CTW algorithm for regression: computing $\mathcal{E}(y_n|z^{n-1}, x_n)$, and updating the tree

Current node: $s_c \leftarrow \varepsilon$, buffer node s_b.
Temporary variables: $x \in \mathbb{R}$, $P_0 \in \mathbb{R}$
Final probability density at y_n, $P \in \mathbb{R}$
1. Reach the deepest node that is relevant for x_n with at least one observation.
while $n_{s_c} > 1$ **do**
  $n_{s_c} \leftarrow n_{s_c} + 1$
  $s_c \leftarrow t(x_n)$, where $t(x_n)$ is the child of s_c such that $x_n \in \mathrm{Dom}(\mathcal{E}_{t(x_n)})$
end while
2. Separate the two observations by going deeper into the tree
$x \leftarrow x_{s_c}$
$s_b \leftarrow s_c$
while $s_b = s_c$ **do**
  $n_{s_c} = 2$
  $s_b \leftarrow t(x)$, where $t(x)$ is the child of s_c such that $x \in \mathrm{Dom}(\mathcal{E}_{t(x)})$
  $s_c \leftarrow t(x_n)$, where $t(x_n)$ is the child of s_c such that $x_n \in \mathrm{Dom}(\mathcal{E}_{t(x_n)})$
end while
$n_{s_b} = 1$; $x_{s_b} = x$
$n_{s_c} = 1$; $x_{s_b} = x_n$
3. Updating the weights, starting from the leaves
$P \leftarrow \mathcal{E}_{s_c}(y_n|z^{n-1}, x_n)$
while $s_c \neq \varepsilon$ **do**
  $s_c \leftarrow \mathrm{father}(s_c)$
  $P_0 \leftarrow P$
  $P \leftarrow w_{s_c}\mathcal{E}_{s_c}(y_n|z^{n-1}, x_n) + (1 - w_{s_c})P$
  Bayesian update of w_s: $w_s \leftarrow \frac{w_s \mathcal{E}_{s_c}(y_n|z^{n-1}, x_n)}{P}$
end while
return P

The generalization to other algorithms (edgewise or using switch) is straightforward.

Appendix B

Geodesic IGO

Proof of Proposition 10.1. Let us first consider the case $k = 1$.

When optimizing a linear function, the non-twisted IGO flow in $\tilde{\mathbb{G}}_d$ with the selection function $w : q \mapsto \mathbf{1}_{q \leqslant q_0}$ is known [OAAH11], and in particular, we have:

$$\mu_t = \mu_0 + \frac{\beta(q_0)}{\alpha(q_0)}\sigma_t, \tag{B.1}$$

$$\sigma_t = \sigma_0 \exp(\alpha(q_0)t), \tag{B.2}$$

where, if we denote by $\mathcal{N}$ a random vector following a standard normal distribution and $\mathcal{F}$ the cumulative distribution of a standard normal distribution,

$$\alpha(q_0, d) = \frac{1}{2d}\left(\int_0^{q_0} \mathcal{F}^{-1}(u)^2 du - q_0\right), \tag{B.3}$$

and:

$$\beta(q_0) = \mathbb{E}(\mathcal{N}\mathbf{1}_{\mathcal{N} \leqslant \mathcal{F}^{-1}(q_0)}). \tag{B.4}$$

In particular, $\alpha := \alpha(\frac{1}{4}, 1) \approx 0.107$ and $\beta := \beta(\frac{1}{4}) \approx -0.319$.

With a minor modification of the proof in [OAAH11], we find that the (η_μ, η_σ)-twisted IGO flow is given by:

$$\mu_t = \mu_0 + \frac{\beta(q_0)}{\alpha(q_0)}\sigma_0 \exp(\eta_\mu \alpha(q_0)t), \tag{B.5}$$

$$\sigma_t = \sigma_0 \exp(\eta_\sigma \alpha(q_0)t), \tag{B.6}$$

Notice that Equation (B.5) shows that the assertions about the convergence of (σ_n) immediately imply the assertions about the convergence of (μ_n).

Let us now consider a step of the GIGO algorithm: The twisted IGO speed is $Y = (\eta_\mu \beta\sigma_0, \eta_\sigma \alpha\sigma_0)$, with $\alpha\sigma_0 > 0$ (*i.e.*, the variance should be increased: this is where we need $q_0 < 0.5$).

Proposition B.5 shows that the covariance at the end of the step is (using the same notation):

$$\sigma(\delta t) = \sigma(0)\Im(\frac{die^{v\delta t} - c}{cie^{v\delta t} + d}) = \sigma(0)\frac{e^{v\delta t}(d^2 + c^2)}{c^2e^{2v\delta t} + d^2} =: \sigma(0)f(\delta t), \tag{B.7}$$

and it is easy to see that f only depends on δt (and on q_0). In other words, $f(\delta t)$ will be the same at each step of the algorithm. The existence of δt_{cr} easily follows (furthermore, recall Figure 8.1 in Section 8.2.1), and δt_{cr} is the positive solution of $f(x) = 1$.

After a quick computation, we find:

$$\exp(v\delta t_{\text{cr}}) = \frac{\sqrt{1+u^2}+1}{\sqrt{1+u^2}-1}. \tag{B.8}$$

where:

$$u := \sqrt{\frac{\eta_\mu}{2n\eta_\sigma}}\frac{\beta}{\alpha}, \tag{B.9}$$

and:

$$v := \sqrt{\eta_\sigma^2\alpha^2 + \frac{\eta_\mu\eta_\sigma}{2n}\beta^2}. \tag{B.10}$$

Finally, for $w = k.\mathbf{1}_{q\leqslant q_0}$, Proposition 9.4 shows that:

$$\delta t_{\text{cr}} = \frac{1}{k}\frac{1}{v}\ln\left(\frac{\sqrt{1+u^2}+1}{\sqrt{1+u^2}-1}\right). \tag{B.11}$$

□

B.1 Generalization of the Twisted Fisher Metric

The following definition is a more general way to introduce the twisted Fisher metric.

Definition B.1. *Let (Θ, g) be a Riemannian manifold, $(\Theta_1, g|_{\Theta_1}), ..., (\Theta_n, g|_{\Theta_n})$, a splitting (as defined in Section 9.1.2) of Θ compatible with the metric g.*

We call $(\eta_1, ..., \eta_n)$-twisted metric on (Θ, g) for the splitting $\Theta_1, ..., \Theta_n$ the metric g' on Θ defined by $g'|_{\Theta_i} = \frac{1}{\eta_i}g|_{\Theta_i}$ for $1 \leqslant i \leqslant n$, and $\Theta_i \perp \Theta_j$ for $i \neq j$.

Proposition B.2. *The (η_μ, η_Σ)-twisted metric on $\mathbb{G}_d$ with the Fisher metric for the splitting $\mathcal{N}(\mu, \Sigma) \mapsto (\mu, \Sigma)$ coincides with the (η_μ, η_Σ)-twisted Fisher metric from Definition 9.1.*

Proof. It is easy to see that the (η_μ, η_Σ)-twisted Fisher metric satisfies the condition in Definition B.1. □

B.2 Twisted Geodesics

The following theorem can be used to compute the twisted geodesics from the non twisted geodesics. It is a simple calculation.

Theorem B.3. *Let $\eta_\mu, \eta_\Sigma \in \mathbb{R}$, $\mu_0 \in \mathbb{R}^d$, $A_0 \in GL_d(\mathbb{R})$, and $(\dot{\mu}_0, \dot{\Sigma}_0) \in T_{\mathcal{N}(\mu_0, A_0 A_0^T)}\mathbb{G}_d$. Let*

$$\begin{array}{rccc} h & : & \mathbb{G}_d & \to & \mathbb{G}_d \\ & & \mathcal{N}(\mu, \Sigma) & \mapsto & \mathcal{N}(\sqrt{\frac{\eta_\mu}{\eta_\Sigma}}\mu, \Sigma) \end{array} . \tag{B.12}$$

We denote by ϕ (resp. ψ) the Riemannian exponential of $\mathbb{G}_d$ (resp. $\mathbb{G}_d$ with the (η_μ, η_Σ)-twisted Fisher metric) at $\mathcal{N}(\sqrt{\frac{\eta_\mu}{\eta_\Sigma}}\mu_0, A_0 A_0^T)$ (resp. $\mathcal{N}(\mu_0, A_0 A_0^T)$). We have:

$$\psi(\dot{\mu}_0, \dot{\Sigma}_0) = h \circ \phi(\sqrt{\frac{\eta_\Sigma}{\eta_\mu}}\dot{\mu}_0, \dot{\Sigma}_0) \tag{B.13}$$

Proof. Let us denote by: $\begin{pmatrix} I_\mu & 0 \\ 0 & I_\Sigma \end{pmatrix}$ the Fisher metric in the parametrization μ, Σ, and consider the following parametrization of $\mathbb{G}_d$: $(\tilde{\mu}, \Sigma) \mapsto \mathcal{N}(\frac{\sqrt{\eta_\Sigma}}{\sqrt{\eta_\mu}}\tilde{\mu}, \Sigma)$.

The Riemannian exponential at $\mathcal{N}(\mu_0, A_0 A_0^T)$ in this parametrization is:

$$h \circ \phi \circ (\mathrm{d}h(\mu_0, A_0 A_0^T))^{-1} \tag{B.14}$$

However, in this parametrization, the Fisher metric reads:

$$\begin{pmatrix} \frac{\eta_\Sigma}{\eta_\mu} I_\mu & 0 \\ 0 & I_\Sigma \end{pmatrix}, \tag{B.15}$$

which is proportional to the (η_μ, η_Σ)-twisted Fisher metric up to a factor $\frac{1}{\eta_\Sigma}$. Consequently, the Christoffel symbols are the same as the Christoffel symbols of the (η_μ, η_Σ)-twisted Fisher metric, and so are the geodesics. Therefore, we have:

$$\psi = h \circ \phi \circ (\mathrm{d}h(\mu_0, A_0 A_0^T))^{-1}, \tag{B.16}$$

which is what we wanted. □

For the remainder of this section, we fix η_μ and η_Σ; $\mathbb{G}_d$ is endowed with the (η_μ, η_Σ)-twisted Fisher metric, and $\tilde{\mathbb{G}}_d$ is endowed with the induced metric. The proofs of the propositions below are a simple rewriting of their non-twisted counterparts that can be found in Sections 8.2 and 8.3 and can be seen as corollaries of Theorem B.3.

Theorem B.4. *If $\gamma\colon t \mapsto \mathcal{N}(\mu(t), \sigma(t)^2 I)$ is a twisted geodesic of $\tilde{\mathbb{G}}_d$, then there exists $a, b, c, d \in \mathbb{R}$, such that $ad - bc = 1$, and $v > 0$, such that $\mu(t) = \mu(0) + \sqrt{\frac{2d\eta_\mu}{\eta_\sigma}} \frac{\dot{\mu}_0}{\|\dot{\mu}_0\|} \tilde{r}(t)$, $\sigma(t) = \Im(\gamma_{\mathcal{C}}(t))$, with $\tilde{r}(t) = \Re(\gamma_{\mathcal{C}}(t))$ and:*

$$\gamma_{\mathcal{C}}(t) := \frac{aie^{vt} + b}{cie^{vt} + d}. \tag{B.17}$$

Proposition B.5. *Let $n \in \mathbb{N}$, $v_\mu \in \mathbb{R}^n$, v_σ, η_μ, η_σ, $\sigma_0 \in \mathbb{R}$, with $\sigma_0 > 0$.*

Let $v_r := \|v_\mu\|$, $\lambda = \sqrt{\frac{2n\eta_\mu}{\eta_\sigma}}$ $v := \sqrt{\frac{\frac{1}{\lambda^2} v_r^2 + v_\sigma^2}{\sigma_0^2}}$, $M_0 := \frac{1}{\lambda} \frac{v_r}{v\sigma_0^2}$ and $S_0 := \frac{v_\sigma}{v\sigma_0^2}$.

Let $c := \left(\frac{\sqrt{M_0^2 + S_0^2} - S_0}{2} \right)^{\frac{1}{2}}$ and $d := \left(\frac{\sqrt{M_0^2 + S_0^2} + S_0}{2} \right)^{\frac{1}{2}}$.

Let $\gamma_{\mathbb{C}}(t) := \sigma_0 \frac{die^{vt} - c}{cie^{vt} + d}$.

Then:

$$\gamma : t \mapsto \mathcal{N}\left(\mu_0 + \lambda \frac{v_\mu}{\|v_\mu\|} \Re(\gamma_{\mathcal{C}}(t)), \Im(\gamma_{\mathcal{C}}(t)) \right) \tag{B.18}$$

is the twisted geodesic of $\tilde{\mathbb{G}}_n$ satisfying $\gamma(0) = (\mu_0, \sigma_0)$ and $\dot{\gamma}(0) = (v_\mu, v_\sigma)$. The regular geodesics of $\tilde{\mathbb{G}}_n$ are obtained with $\eta_\mu = \eta_\sigma = 1$.

Theorem B.6. *Let $\gamma : t \mapsto \mathcal{N}(\mu_t, \Sigma_t)$ be a twisted geodesic of $\mathbb{G}_d$. Then, the following quantities are invariant:*

$$J_\mu = \frac{1}{\eta_\mu} \Sigma_t^{-1} \dot{\mu}_t, \tag{B.19}$$

$$J_\Sigma = \Sigma_t^{-1} \left(\frac{1}{\eta_\mu} \dot{\mu}_t \mu_t^T + \frac{1}{\eta_\Sigma} \dot{\Sigma}_t \right). \tag{B.20}$$

Theorem B.7. *If $\mu : t \mapsto \mu_t$ and $\Sigma : t \mapsto \Sigma_t$ satisfy the equations:*

$$\dot{\mu}_t = \eta_\mu \Sigma_t J_\mu \tag{B.21}$$

$$\dot{\Sigma}_t = \eta_\Sigma \Sigma_t (J_\Sigma - J_\mu \mu_t^T) = \eta_\Sigma \Sigma_t J_\Sigma - \frac{\eta_\Sigma}{\eta_\mu} \dot{\mu}_t \mu_t^T, \tag{B.22}$$

where:

$$J_\mu = \frac{1}{\eta_\mu} \Sigma_0^{-1} \dot{\mu}_0,$$

and:

$$J_\Sigma = \Sigma_0^{-1} \left(\frac{1}{\eta_\mu} \dot{\mu}_0 \mu_0^T + \frac{1}{\eta_\Sigma} \dot{\Sigma}_0 \right).$$

then $t \mapsto \mathcal{N}(\mu_t, \Sigma_t)$ is a twisted geodesic of $\mathbb{G}_d$.

Theorem B.8. *If $\mu : t \mapsto \mu_t$ and $A : t \mapsto A_t$ satisfy the equations:*

$$\dot{\mu} = \eta_\mu A_t A_t^T J_\mu, \tag{B.23}$$

$$\dot{A}_t = \frac{\eta_\Sigma}{2}(J_\Sigma - J_\mu \mu_t^T)^T A_t, \tag{B.24}$$

where:

$$J_\mu = \frac{1}{\eta_\mu}(A_0^{-1})^T A_0^{-1}\dot{\mu}_0$$

and:

$$J_\Sigma = (A_0^{-1})^T A_0^{-1}(\frac{1}{\eta_\mu}\dot{\mu}_0\mu_0^T + \frac{1}{\eta_\Sigma}\dot{A}_0 A_0^T + \frac{1}{\eta_\Sigma}A_0\dot{A}_0^T),$$

then $t \mapsto \mathcal{N}(\mu_t, A_t A_t^T)$ *is a twisted geodesic of* $\mathbb{G}_d$.

B.3 Pseudocodes

B.3.1 For All Algorithms

All studied algorithms have a common part, given here:

Variables: μ, Σ (or A such that $\Sigma = AA^T$).

List of parameters: $f\colon \mathbb{R}^d \to \mathbb{R}$, step size δt, learning rates η_μ, η_Σ, sample size λ, weights $(w_i)_{i\in[1,\lambda]}$, N number of steps for the Euler method, r Euler step size reduction factor (for GIGO-Σ only).

Algorithm 3 For all algorithms.

$\mu \leftarrow \mu_0$
if The algorithm updates Σ directly **then**
  $\Sigma \leftarrow \Sigma_0$
  Find some A, such that $\Sigma = AA^T$
else {The algorithm updates a square root A of Σ}
  $A \leftarrow A_0$
  $\Sigma = AA^T$
end if
while NOT (Termination criterion) **do**
  for $i = 1$ to λ **do**
    $z_i \sim \mathcal{N}(0, I)$
    $x_i = Az_i + \mu$
  end for
  Compute the IGO initial speed, and update the mean and the covariance (the updates are Algorithms 4 to 8).
end while

Notice that we always need a square root A of Σ to sample the x_i, but the decomposition $\Sigma = AA^T$ is not unique. Two different decompositions will give two algorithms, such that one is a modification of the other as a stochastic process: same law (the x_i are abstractly sampled from $\mathcal{N}(\mu, \Sigma)$), but different trajectories (for given z_i, different choices for the square root will give different x_i). For GIGO-Σ, since we have to invert the covariance matrix, we used the Cholesky decomposition (A lower triangular. The the other implementation directly maintains a square root of Σ). Usually, in CMA-ES, the square root of Σ ($\Sigma = AA^T$, A symmetric) is used.

B.3.2 Updates

When describing the different updates, μ, Σ, A, the x_i and the z_i are those defined in Algorithm 3.

For Algorithm 4 (GIGO-Σ), when the covariance matrix after one step is not positive-definite, we compute the update again, with a step size divided by r for the Euler method (we have no reason to recommend any particular value of r, the only constraint is $r > 1$).

Algorithm 4 GIGO Update, one step, updating the covariance matrix.

1. Compute the IGO speed:

$$v_\mu = A \sum_{i=1}^{\lambda} w_i z_i,$$

$$v_\Sigma = A \sum_{i=1}^{\lambda} w_i \left(z_i z_i^T - I \right) A^T.$$

2. Compute the Noether invariants:

$J_\mu \leftarrow \Sigma^{-1} v_\mu$,
$J_\Sigma \leftarrow \Sigma^{-1} ({v_\mu}^t \mu + v_\Sigma)$.

3. Solve numerically the equations of the geodesics:

Unhappy $\leftarrow$ true
$\mu_0 \leftarrow \mu$
$\Sigma_0 \leftarrow \Sigma$
$k = 0$
while Unhappy **do**
  $\mu \leftarrow \mu_0$
  $\Sigma \leftarrow \Sigma_0$
  $h \leftarrow \delta t / (N r^k)$
  for $i = 1$ to $N r^k$ **do**
    $\mu \leftarrow \mu + h \eta_\mu \Sigma J_\mu$
    $\Sigma \leftarrow \Sigma + h \eta_\Sigma \Sigma (J_\Sigma - J_\mu \mu^T)$
  end for
  if Σ positive-definite **then**
    Unhappy $\leftarrow$ false
  end if
  $k \leftarrow k + 1$
end while
return μ, Σ

Algorithm 5 GIGO Update, one step, updating a square root of the covariance matrix.

1. Compute the IGO speed:

$$v_\mu = A \sum_{i=1}^{\lambda} w_i z_i,$$

$$v_\Sigma = A \sum_{i=1}^{\lambda} w_i \left(z_i z_i^T - I \right) A^T.$$

2. Compute the Noether invariants:
$J_\mu \leftarrow \Sigma^{-1} v_\mu$,
$J_\Sigma \leftarrow \Sigma^{-1} ({v_\mu}^t \mu + v_\Sigma)$.

3. Solve numerically the equations of the geodesics:
$h \leftarrow \delta t / N$
for $i = 1$ to N **do**
  $\mu \leftarrow \mu + h \eta_\mu A A^T J_\mu$
  $A \leftarrow A + \frac{h}{2} \eta_\Sigma (J_\Sigma - J_\mu \mu^T)^T A$
end for
return μ, A

Algorithm 6 Exact GIGO, one step. Not exactly our implementation; see the discussion after Corollary 8.17.

1. Compute the IGO speed:

$$v_\mu = A \sum_{i=1}^{\lambda} w_i z_i,$$

$$v_\Sigma = A \sum_{i=1}^{\lambda} w_i \left(z_i z_i^T - I \right) A^T.$$

2. Learning rates
$\lambda \leftarrow \sqrt{\frac{\eta_\Sigma}{\eta_\mu}}$
$\mu \leftarrow \lambda \mu$
$v_\mu \leftarrow \eta_\mu \lambda v_\mu$
$v_\Sigma \leftarrow \eta_\Sigma v_\Sigma$
3. Intermediate computations.
$G^2 \leftarrow A^{-1} \left(v_\Sigma (A^{-1})^T A^{-1} v_\Sigma + 2 v_\mu v_\mu^T \right) (A^{-1})^T$
$C_1 \leftarrow \mathrm{ch}(\frac{G}{2})$
$C_2 \leftarrow \mathrm{sh}(\frac{G}{2}) G^{-1}$
$R \leftarrow \left((C_1 - A^{-1} v_\Sigma (A^{-1})^T C_2)^{-1} \right)^T$
4. Actual update
$\mu \leftarrow \mu + 2 A R C_2 A^{-1} v_\mu$
$A \leftarrow AR$
5. Return to the "real" μ
$\mu \leftarrow \frac{\mu}{\lambda}$
return μ, A

Algorithm 7 xNES update, one step.

1. Compute G_μ and G_M (equivalent to the computation of the IGO speed):

$$G_\mu = \sum_{i=1}^{\lambda} w_i z_i$$

$$G_M = \sum_{i=1}^{\lambda} w_i \left(z_i z_i^T - I \right)$$

2. Actual update:

$\mu \leftarrow \mu + \eta_\mu A G_\mu$

$A \leftarrow A + A \exp(\eta_\Sigma G_M / 2)$

return μ, A

Algorithm 8 pure rank-μ CMA-ES update, one step

1. Computation of the IGO speed:

$$v_\mu = \sum_{i=1}^{\lambda} w_i (x_i - \mu)$$

$$v_\Sigma = \sum_{i=1}^{\lambda} w_i \left((x_i - \mu)(x_i - \mu)^T - \Sigma \right)$$

2. Actual update:

$\mu \leftarrow \mu + \eta_\mu v_\mu$

$\Sigma \leftarrow \Sigma + \eta_\Sigma v_\Sigma$

return μ, Σ

Algorithm 9 GIGO in $\tilde{\mathbb{G}}_d$, one step.

1. Compute the IGO speed:

$$Y_\mu = \sum_{i=1}^{\lambda} w_i (x_i - \mu) \; ; \; Y_\sigma = \sum_{i=1}^{\lambda} w_i \left(\frac{(x_i - \mu)^T (x_i - \mu)}{2d\sigma} - \frac{\sigma}{2} \right)$$

2. Better parametrization:

$$\lambda := \sqrt{\frac{2d\eta_\mu}{\eta_\sigma}}$$

$$v_r := \frac{\eta_\mu}{\lambda} \|Y_\mu\| \; ; \; v_\sigma := \eta_\sigma Y_\sigma$$

3. Find a, b, c, d, v corresponding to $\mu, \sigma, \dot{\mu}, \dot{\sigma}$:

$$v = \sqrt{\frac{v_r^2 + v_\sigma^2}{\sigma^2}}$$

$$S_0 := \frac{v_\sigma}{v\sigma^2} \; ; \; M_0 := \frac{v_r}{v\sigma^2}$$

$$C := \frac{\sqrt{S_0^2 + M_0^2} - S_0}{2} \; ; \; D := \frac{\sqrt{S_0^2 + M_0^2} + S_0}{2}$$

$$c := \sqrt{C} \; ; \; d := \sqrt{D}$$

4. Actual Update:

$$z := \sigma \frac{die^{v\delta t} - c}{cie^{v\delta t} + d}$$

$$\mu := \mu + \lambda \Re(z) \frac{Y_\mu}{\|Y_\mu\|} \; ; \; \sigma := \Im(z)$$

return μ, σ

Bibliography

[Aka73] Hirotogu Akaike. Information theory and an extension of the maximum likelihood principle. In *Selected Papers of Hirotugu Akaike*, pages 199–213. Springer, 1973.

[Aka09] Nathalie Akakpo. *Adaptive estimation by selecting a best partition into dyadic rectangles*. Theses, Université Paris Sud - Paris XI, December 2009. Rapporteurs :
 Fabienne Comte (Université Paris Descartes)
 Enno Mammen (Université de Mannheim)
.

[AMS08] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

[AN07] S.I. Amari and H. Nagaoka. *Methods of Information Geometry*. Translations of Mathematical Monographs. American Mathematical Society, 2007.

[ANOK10] Youhei Akimoto, Yuichi Nagata, Isao Ono, and Shigenobu Kobayashi. Bidirectional relation between CMA evolution strategies and natural evolution strategies. *Proceedings of Parallel Problem Solving from Nature*, 2010.

[AO13] Youhei Akimoto and Yann Ollivier. Objective improvement in information-geometric optimization. *FOGA 2013*, 2013.

[AVW89] V.I. Arnold, K. Vogtmann, and A. Weinstein. *Mathematical Methods of Classical Mechanics*. Graduate Texts in Mathematics. Springer, 1989.

[BC95] Shumeet Baluja and Rich Caruana. Removing the genetics from the standard genetic algorithm. pages 38–46. Morgan Kaufmann Publishers, 1995.

[Ben15] Jérémy Bensadon. Black-box optimization using geodesics in statistical manifolds. *Entropy*, 17(1):304, 2015.

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA, 1984.

[Bou07] J.P. Bourguignon. *Calcul variationnel*. Ecole Polytechnique, 2007.

[Cha71] Gregory Chaitin. Computational complexity and gödel's incompleteness theorem. *ACM SIGACT News*, 1971.

[CO91] Miquel Calvo and Josep Maria Oller. An explicit solution of information geodesic equations for the multivariate normal model. *Statistics & Decisions 9*, 1991.

[CV] Rudi Cilibrasi and Paul Vitanyi. Clustering by compression.

[Daw84] A.P. Dawid. Statistical theory: the prequential approach (with discussion). *J. R. Statist. Soc. A*, 147:278–292, 1984.

[DSG+11] Wierstra. D., T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. Natural evolution strategies. *CoRR*, abs/1211.0587, 2011.

[EKH10] Henning P. Eberhardt, Vesa Klumpp, and Uwe D. Hanebeck. Density trees for efficient nonlinear state estimation. In *FUSION*, pages 1–8. IEEE, 2010.

[Eli75] Peter Elias. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, 21(2):194–203, 1975.

[Eri87] P Eriksen. Geodesics connected with the fisher metric on the multivariate normal manifold. *Proceedings of the GST Workshop, Lancaster*, 1987.

[Fan61] Robert M. Fano. Transmission of information a statistical theory of communications, 1961.

[Fin97] Daniel Fink. A compendium of conjugate priors, 1997.

[Fis22] R.A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London*, 1922.

[FSSW97] Yoav Freund, Robert E. Schapire, Yoram Singer, and Manfred K. Warmuth. Using and combining predictors that specialize. In Frank Thomson Leighton and Peter W. Shor, editors, *STOC*, pages 334–343. ACM, 1997.

[GHL04] S. Gallot, D. Hulin, and J. LaFontaine. *Riemannian Geometry*. Universitext (1979). Springer-Verlag GmbH, 2004.

[Gru07] Peter D. Grunwald. *The Minimum Description Length Principle (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.

[GSY+10] Tobias Glasmachers, Tom Schaul, Sun Yi, Daan Wierstra, and Jurgen Schmidhuber. Exponential natural evolution strategies. *GECCO*, 2010.

[Han11] Nikolaus Hansen. The CMA evolution strategy: A tutorial. 2011.

[HTF01] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[Hua13] Wen Huang. *Optimization algorithms on Riemannian manifolds with applications*. PhD thesis, Florida state university, 2013.

[Huf52] David A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the Institute of Radio Engineers*, 40(9):1098–1101, September 1952.

[ITW11] Takuro Imai, Akira Takaesu, and Masato Wakayama. Remarks on geodesics for multivariate normal models. *Journal of Math-for-Industry*, 2011.

[Jef61] H. Jeffreys. *Theory of Probability*. Oxford, Oxford, England, third edition, 1961.

[JLJ98] J. Jost and X. Li-Jost. *Calculus of Variations*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1998.

[KAW12] Wouter M. Koolen, Dmitry Adamskiy, and Manfred K. Warmuth. Putting bayes to sleep. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *NIPS*, pages 135–143, 2012.

[KdR08] Wouter M. Koolen and Steven de Rooij. Combining expert advice efficiently. *CoRR*, abs/0802.2015, 2008.

[KM07] Petri Kontkanen and Petri Myllymäki. Mdl histogram density estimation. In *In Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (to appear*, 2007.

[KMH+03] Stefan Kern, Sibylle D. Müller, Nikolaus Hansen, Dirk Büche, Jiri Ocenasek, and Petros Koumoutsakos. Learning probability distributions in continuous evolutionary algorithms - a comparative review. *Natural Computing*, 3:77–112, 2003.

[Kol65] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.

[KT81] R. Krichevsky and V. Trofimov. The performance of universal encoding. *Information Theory, IEEE Transactions on*, 27(2):199–207, Mar 1981.

[LV08] Ming Li and Paul M.B. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Publishing Company, Incorporated, 3 edition, 2008.

[MAJ+98] J. S. Marron, S. Adak, I. M. Johnstone, M. H. Neumann, and P. Patil. Exact risk analysis of wavelet regression. *Journal of Computational and Graphical Statistics*, 7(3):278–309, 1998.

[MMP11] Luigi Malagò, Matteo Matteucci, and Giovanni Pistone. Towards the geometry of estimation of distribution algorithms based on the exponential family. In Hans-Georg Beyer and William B. Langdon, editors, *FOGA*, pages 230–242. ACM, 2011.

[MP14] Luigi Malagò and Giovanni Pistone. Combinatorial optimization with information geometry: The newton method. *Entropy*, 16(8):4260–4289, 2014.

[OAAH11] Yann Ollivier, Ludovic Arnold, Anne Auger, and Nikolaus Hansen. Information-geometric optimization algorithms : A unifying picture via invariance principles. *Preprint*, 2011.

[OHSS12] Alexander O'Neill, Marcus Hutter, Wen Shao, and Peter Sunehag. Adaptive context tree weighting. *CoRR*, abs/1201.2056, 2012.

[PF86] Boaz Porat and Benjamin Friedlander. Computation of the exact information matrix of Gaussian time series with stationary random components. *IEEE Transactions on acoustics speech and signal processing*, February 1986.

[RG11] Parikshit Ram and Alexander G. Gray. Density estimation trees. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 627–635, New York, NY, USA, 2011. ACM.

[Ris76] Jorma Rissanen. Generalized kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, 20(3):198–203, 1976.

[Ris78] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[RK05] Mohammad M. Rashid and Tsutomu Kawabata. Analysis of zero-redundancy estimator with a finite window for markovian source. *IEICE Transactions*, 88-A(10):2819–2825, 2005.

[RR07] Jorma Rissanen and Teemu Roos. Conditional NML universal models. In *Information Theory and Applications Workshop, 2007*, pages 337–341, Jan 2007.

[Sch78] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[Sha48] Claude E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[Sha51] Claude Elwood Shannon. Prediction and entropy of printed english. *Bell System Technical Journal*, 30:50–64, January 1951.

[Sol64] Ray J. Solomonoff. A formal theory of inductive inference. *Information and Control*, 7, 1964.

[SSUSCU81] Lene Theil Skovgaard and 005 STANFORD UNIVERSITY. Stanford (CA US). A riemannian geometry of the multivariate normal model, 1981.

[vE10] Tim van Erven. *When Data Compression and Statistics Disagree*. PhD thesis, Centrum voor Wiskunde en Informatica, 2010.

[vEGdR12] Tim van Erven, Peter Grünwald, and Steven de Rooij. Catching up faster by switching sooner: a predictive approach to adaptive estimation with an application to the AIC–BIC dilemma. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2012.

[VNHB11] Joel Veness, Kee Siong Ng, Marcus Hutter, and Michael H. Bowling. Context tree switching. *CoRR*, abs/1111.3182, 2011.

[VWBG12] Joel Veness, Martha White, Michael Bowling, and András György. Partition tree weighting. *CoRR*, abs/1211.0587, 2012.

[Wil94] Frans M. J. Willems. The context-tree weighting method: Extensions. *IEEE Transactions on Information Theory*, 44:792–798, 1994.

[WSG+14] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15:949–980, 2014.

[WST95] Frans M. J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. The context tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41:653–664, 1995.

[ZL70] A.K. Zvonkin and L.A. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, 1970.

Index

Applications of Information Theory to Machine Learning

Summary: We study two different topics, using insight from information theory in both cases: – Context Tree Weighting is a text compression algorithm that efficiently computes the Bayesian combination of all visible Markov models: we build a "context tree", with deeper nodes corresponding to more complex models, and the mixture is computed recursively, starting with the leaves. We extend this idea to a more general context, also encompassing density estimation and regression; and we investigate the benefits of replacing regular Bayesian inference with switch distributions, which put a prior on sequences of models instead of models. – Information Geometric Optimization (IGO) is a general framework for black box optimization that recovers several state of the art algorithms, such as CMA-ES and xNES. The initial problem is transferred to a Riemannian manifold, yielding parametrization-invariant first order differential equation. However, since in practice, time is discretized, this invariance only holds up to first order. We introduce the Geodesic IGO (GIGO) update, which uses this Riemannian manifold structure to define a fully parametrization invariant algorithm. Thanks to Noether's theorem, we obtain a first order differential equation satisfied by the geodesics of the statistical manifold of Gaussians, thus allowing to compute the corresponding GIGO update. Finally, we show that while GIGO and xNES are different in general, it is possible to define a new "almost parametrization-invariant" algorithm, Blockwise GIGO, that recovers xNES from abstract principles.

Keywords: MDL, Switching, Prediction, Black-box optimization, Riemannian geometry

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Applications de la théorie de l'information à l'apprentissage statistique

Résumé: On considère ici deux sujets différents, en utilisant des idées issues de la théorie de l'information:

– Context Tree Weighting est un algorithme de compression de texte qui calcule exactement une prédiction Bayésienne qui considère tous les modèles markoviens visibles: on construit un "arbre de contextes", dont les nœuds profonds correspondent aux modèles complexes, et la prédiction est calculée récursivement à partir des feuilles. On étend cette idée à un contexte plus général qui comprend également l'estimation de densité et la régression, puis on montre qu'il est intéressant de remplacer les mixtures Bayésiennes par du "switch", ce qui revient à considérer a priori des suites de modèles plutôt que de simples modèles.

– Information Geometric Optimization (IGO) est un cadre général permettant de décrire plusieurs algorithmes d'optimisation boîte noire, par exemple CMA-ES et xNES. On transorme le problème initial en un problème d'optimisation d'une fonction lisse sur une variété Riemannienne, ce qui permet d'obtenir une équation différentielle du premier ordre invariante par reparamétrage. En pratique, il faut discrétiser cette équation, et l'invariance n'est plus valable qu'au premier ordre. On définit l'algorithme IGO géodésique (GIGO), qui utilise la structure de variété Riemannienne mentionnée ci-dessus pour obtenir un algorithme totalement invariant par reparamétrage. Grâce au théorème de Noether, on obtient facilement une équation différentielle du premier ordre satisfaite par les géodésiques de la variété statistique des gaussiennes, ce qui permet d'implémenter GIGO. On montre enfin que xNES et GIGO sont différents dans le cas général, mais qu'il est possible de définir un nouvel algorithme presque invariant par reparamétrage, GIGO par blocs, qui correspond exactement à xNES dans le cas Gaussien.

Mots clés: MDL, Switching, Prediction, Optimisation boîte noire, Géométrie riemannienne

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France