

Information theory and inference

Simple paths to survive the course

by Giulia Campesan, Filippo Conforto, Tommaso Faorlin,

Alessandro Marcomini, Lorenzo Rosset, Andrea Zanola.

We are six students enrolled in the Physics of Data master degree at the University of Padova, Italy. From this year on, our master provides lectures on *Information Theory and Inference*, held by two international expert scientists that have been working on these topics for several years. Since the arguments tackled seemed very interesting, we decided to put the efforts together and work on some high quality notes for the whole course, to allow the other students to study and have a different point of view on the various arguments treated. We hope you will enjoy our vegetable soup with melt cheese.

Giulia Campesan
Filippo Conforto
Tommaso Faorlin
Alessandro Marcomini
Lorenzo Rosset
Andrea Zanola

Padova, August 7, 2021

August 7, 2021
Version 1.0

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Contents

I	Carlo Albert	1
1	Basics Principles	3
1.1	Bayesian Statistics	3
1.1.1	The setup	3
1.1.2	Step 1: Prior	4
1.1.3	Step 2: Calibration	4
1.1.4	Step 3: Probabilistic predictions	5
1.2	Monte Carlo methods	5
1.2.1	General building blocks of samplers	6
2	Metropolis algorithms	9
2.1	Markov Chain Monte Carlo	9
2.1.1	From global to local sampling	9
2.1.2	The problem of correlations	12
2.2	Metropolis algorithm	14
2.3	Metropolis algorithm tuning	17
2.3.1	Haario algorithm	17
2.3.2	Vihola algorithm	19
2.3.3	EMCEE sampler	20
2.4	Gibbs Sampling	22
3	Hamiltonian Monte Carlo	25
3.1	Basic Concepts	25
3.2	Advanced Hamiltonian Monte Carlo	27
3.2.1	Example: Stochastic differential equation (SDE) model	27
3.2.2	Riemann Manifold Hamiltonian Monte Carlo	31

4	Approximate Bayesian Computation	37
4.1	Basic concepts	37
4.1.1	Summary statistics: basic idea	38
4.2	Tolerance: the SABC algorithm	38
4.2.1	Adaptive schedule	41
4.2.2	SABC tunable parameters	45
4.3	Summary statistics	46
4.3.1	The exponential family	47
4.3.2	Phases and phase transitions in inference problems	48
5	ML-approaches to Bayesian Inference	53
5.1	Introduction	53
5.2	The variational Bayes method	53
5.3	ML alternatives to ABC - I	55
5.4	ML alternatives to ABC - II	57
II	Jeff Byers	59
6	The Bayesian approach	61
6.1	On the Bayesian interpretation	61
6.1.1	Physics as Encoding, Decoding and Bottlenecks	61
6.1.2	The Bayes' Theorem	62
6.1.3	What is probability?	63
6.1.4	From sets to space of models	64
6.1.5	Parameters estimation: Gull's problem solution	65
6.1.6	Parameters estimation: Bretthorst' spectral analysis	66
6.2	Conjugate priors	67
6.2.1	Discrete distributions: solving the Coin Tossing problem	67
6.2.2	Continuous distributions: Gaussian-shaped likelihood	69
6.2.3	Predictive posteriors	70
7	Entropy and Information	73
7.1	Learning by diffusing: the information potential	73
7.2	Distance measure in information theory	74
7.2.1	Self-information or Information Potential	75
7.2.2	Decomposition of the Entropy	75
7.2.3	Joint entropy and conditional entropy	77
7.2.4	Kullback-Leibler Divergence or Relative Entropy	77
7.2.5	Alternative measure functions	78
7.2.6	Mutual information	79
7.3	Information theory version of Bayes' rule	81
8	Model Comparison	83
8.1	Occam's razor	83
8.1.1	Model comparison and Occam's razor	83
8.1.2	Evidence and the Occam's factor	84

8.1.3	The big picture	85
8.2	Creating models, making choices and Bayesian inference	86
8.3	Language model and distance in the space of parameters	89
8.3.1	Using Fisher information: the Fisher scoring algorithm	90
8.4	Information Geometry	91
8.4.1	Riemannian Manifold	91
8.4.2	Example: 1-D Gaussian pdf	93
8.4.3	Connection with Inference	95
8.5	Exercise solutions	97
8.5.1	Exercise 28.4 (Mac03)	97
9	Communications channel	101
9.1	Communications channels and information transmission	102
9.1.1	Communication models	102
9.2	Binary classifiers as Binary Asymmetric Channels	103
9.2.1	Evaluating Binary Classifiers	104
9.3	Relevance	105
9.3.1	Relevance in information theory and data analysis	106
9.3.2	Information bottleneck	108
9.4	Machine learning	108
9.4.1	Statistics versus machine learning	108
9.4.2	Labels	110
9.4.3	The relationship of machine learning to Bayesian inference	111
9.5	Predictive information	113
9.5.1	Mutual information between the Past and the Future	114
9.5.2	Determine I_{pred} for a Markov Process	116
10	Non parametric models	117
10.1	Gaussian process	117
10.1.1	Nonlinear regression: parametric approach	117
10.1.2	From parametric models to Gaussian processes	118
10.1.3	Using a given Gaussian process model in regression	120
10.2	The intuition behind Gaussian process and kernels	122
10.2.1	Gram's matrix and covariance matrix	122
10.2.2	Kernel as local average	123
10.3	Diffusion on a manifold	123
10.3.1	How to infer a generic kernel	124
10.4	Dirichlet process	125
10.4.1	The Dirichlet distribution	125
10.4.2	Paper discussion about PDFs of probabilities	127
10.4.3	The Dirichlet process	128

Part I

Carlo Albert



1. Basics Principles

1.1 Bayesian Statistics

The paradigm behind Bayesian statistics is to express knowledge (or belief) about any kind of variable in the form of a probability distribution. A priori, though, it might not be obvious why we should use probability theory to express some subjective knowledge about variables. One possible explanation goes back to the De Finetti claim that if we want to find out what a particular person thinks about the value of a particular variable, we can operationalize this by means of lotteries. Let's see an example. Lesson 1
22/03
LR

Suppose to play a game in which a dice is tossed and you have to bet on the outcome. You have to choose between two lotteries:

- **L1**: if the outcome is 6 you win 1000€, otherwise you have to pay 1€;
- **L2**: if the outcome is 6 you pay 1000€, otherwise you win 1€

Which of the two alternatives would you take? Unless you're crazy, it's obvious that the right choice is L1, and the reason is that the win/loss ratio is much bigger than the corresponding odds ratio

$$\frac{1000}{1} \gg \frac{1-p}{p} = \frac{5}{1}$$

The idea here is that when the prizes are such that a person is indifferent about L1 or L2, the win/loss ratio corresponds to the odds ratio he assigns to the event, from which a probability can be derived. For example, if you ask many people what they think about the lottery

- **L3**: if the outcome is 6 you win 1000€, otherwise you have to pay 250€;

then, on average, the result will be quite balanced. The *operational subjective probability* can be therefore derived by setting the prizes of the lottery in such a way to make people indifferent about the bet to be made.

Under the assumption that that person is rational and wants to avoid sure loss, one can eventually retrieve the **Kolmogorov axioms of probability theory**: given an event space Ω , and given two random events $A, B \subset \Omega$, we have

1. $p(A) \geq 0$
2. $p(\Omega) = 1$
3. $p(A \cup B) = p(A) + p(B)$ if $A \cap B = \emptyset$

This is the way De Finetti argues why we should use probability theory to express subjective belief.

1.1.1 The setup

In the Bayesian model based data analysis, we're interested in three kind of variables:

- **x**: input variables (known)
- **y**: output variables (to be observed)
- **θ** : parameters (to be inferred)

Given the input data, the output and the model we want to exploit to describe the phenomenon, the task of Bayesian inference is to calibrate the model's parameters in order to have the best agreement with observations.

As an example, consider the rainfall-runoff analysis of a river (see Figure 1.1). We can model the river as a cascade of reservoirs (tanks, or pockets), in which the storage, the outflow and the runoff of each reservoir is governed by differential equations with different parameters. We aim to use our knowledge (data and observations) about the real world process in order to tune such parameters and to get the best possible description of the river's behaviour, in such a way to make predictions based on rainfall forecasts.

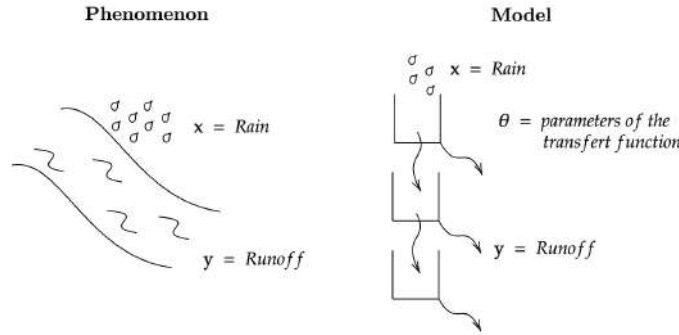


Figure 1.1: Model of the rainfall-runoff of a river

Bayesian inference can be summarized in three main steps.

1.1.2 Step 1: Prior

In general, from the basic laws of probability theory we can write the joint probability density of observations \mathbf{y} and model's parameters $\boldsymbol{\theta}$ given the input data \mathbf{x} as

$$f(\mathbf{y}, \boldsymbol{\theta} | \mathbf{x}) = f(\mathbf{y} | \boldsymbol{\theta}, \mathbf{x}) f(\boldsymbol{\theta}) \quad (1.1)$$

The rationale behind this formulation is that we believe that the observations, \mathbf{y}_{obs} , are a realization from the distribution $f(\mathbf{y} | \boldsymbol{\theta}^*, \mathbf{x})$, for fixed but unknown parameters $\boldsymbol{\theta}^*$. $f(\boldsymbol{\theta})$ express our prior knowledge about parameters, which can come from previous measurements or even heuristic assumptions we make; for this reason it's called **prior** pdf. If a lot of prior knowledge is available, we can make predictions directly using this prior (e.g. models based on first principles in physics). In many disciplines, prior knowledge does not suffice to make predictions, but we need to calibrate the models first, i.e. constrain $\boldsymbol{\theta}$ based on \mathbf{y}_{obs} .

In Eq. (1.1) we can recognize two kinds of uncertainty: $f(\mathbf{y} | \boldsymbol{\theta}, \mathbf{x})$ represents the *aleatory uncertainty* coming from the intrinsic system's randomness, whereas $f(\boldsymbol{\theta})$ brings what is called *epistemic uncertainty*, which means that the "error" is influenced by our prior knowledge about the phenomenon. However, most of times the distinction between the two is not that sharp.

For what concerns models, we can distinguish them into two categories:

- **Mechanistic models:** they are specific of the system, and very often the parameters $\boldsymbol{\theta}$ are interpretable as physical quantities. This gives us also an intuition on what's going on.
- **Machine learning (ML) models:** here the model is not related to the particular system, but it's an architecture (neural network or another) made to predict the outcomes or recognize features of the data. Typically the parameters have no physical meaning at all.

Most of times we deal with something in between.

1.1.3 Step 2: Calibration

Of course, an important step in our Bayesian process is how we use data. Basically, calibration means that we want to condition our prior on the data \mathbf{y}_{obs} we collect. The result of this operation is the so called **posterior** pdf, namely:

$$f(\boldsymbol{\theta} | \mathbf{y}_{obs}, \mathbf{x}) = \frac{f(\mathbf{y}_{obs} | \boldsymbol{\theta}, \mathbf{x}) f(\boldsymbol{\theta})}{\int f(\mathbf{y}_{obs} | \boldsymbol{\theta}, \mathbf{x}) f(\boldsymbol{\theta}) d\boldsymbol{\theta}} \quad (1.2)$$

where $f(\mathbf{y}_{obs}|\boldsymbol{\theta}, \mathbf{x})$ is the **likelihood** function and the denominator is the marginalized pdf $f(\mathbf{y}_{obs}, \mathbf{x})$ (called *evidence*) and it will play a fundamental role in comparing models. However, this last term is just a normalization constant, and typically we don't want to compute it because this would be very expensive: that's why MC methods were introduced. This learning process can be iterated every time we collect more data, and eventually we get a posterior function conditioned by all our set of observations: $f(\boldsymbol{\theta}|\mathbf{y}_{obs,1}, \dots, \mathbf{y}_{obs,n}, \mathbf{x})$.

Modelling not only means to express what we know about a system, but also what we don't know. Consider the fact that, since we're dealing with probability distributions, the outcome of our inference will be a random variable having its characteristic spread. Traditionally, one approach is to write the probabilistic model as

$$\mathbf{Y}(\boldsymbol{\theta}, \mathbf{x}) = \mathbf{y}_{det}(\boldsymbol{\theta}, \mathbf{x}) + E(\boldsymbol{\theta}, \mathbf{x}),$$

where \mathbf{Y} is a random variable corresponding to the pdf $f(\mathbf{y}|\boldsymbol{\theta})$. The *deterministic* term is the result of a PDE system describing the phenomena, whereas $E(\boldsymbol{\theta}, \mathbf{x})$ is an additive uncertainty called *error modelling*. This paradigm has severe shortcomings, though. First of all it's difficult to find an appropriate distribution that is able to represent the intrinsic variability of data, because often the uncertainty is not on the output but it's somewhere else (for example in the input, due to some limits of our experimental apparatus). Moreover, the calibrated model can only be used to predict the output components for which we have data for calibration. The right way to proceed would be to introduce the uncertainty where it arises, but unfortunately this can lead to intractable likelihood functions.

1.1.4 Step 3: Probabilistic predictions

After the model has been calibrated, we can use it to make probabilistic predictions, using the posterior as calibration function:

$$f_{pred}(\mathbf{y}|\mathbf{x}) = \int f_{pred}(\mathbf{y}|\boldsymbol{\theta}, \mathbf{x}) f_{cal}(\boldsymbol{\theta}|\mathbf{y}_{obs}) d\boldsymbol{\theta} \quad (1.3)$$

The prediction and calibration models are often the same, but they can also be different and share the same parameters and inputs. All these integrals are often in high dimensions, and the standard numerical methods for integrating functions would not be able to accomplish the task. This is one of the reasons we need to introduce Monte Carlo methods (for some more details about this formula see next chapters).

1.2 Monte Carlo methods

Monte Carlo methods have been introduced in the Manhattan project, when physicists and engineers realized that hard problems can be solved using randomness. For example, suppose that we want to compute the value of π up to a certain precision. In order to do that we can exploit its series expansion:

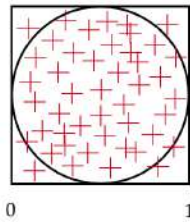
$$\frac{\pi}{4} = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \sum_{n=0}^N \frac{(-1)^n}{2n+1} + \mathcal{O}(1/N)$$

where the last term is a bias that can be reduced by pushing further and further the computation.

Finding π with the Monte Carlo method, instead, consists on sampling points uniformly inside the unit square and to compute the fraction of them that falls inside the unit circle. If we define

$$Z = \#\{x_i^2 + y_i^2 \leq 1 : x_i \sim \mathcal{U}[0, 1], y_i \sim \mathcal{U}[0, 1], i = 1, \dots, N\},$$

then we have $\langle Z/N \rangle = \pi/4$ with no bias, but with an uncertainty $\sqrt{\text{Var}(Z/N)} \sim \mathcal{O}(1/\sqrt{N})$.



In general, MC can be used to compute expectation values

$$\langle h \rangle_{\mathbf{x}} = \int h(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} \approx \frac{1}{N} \sum_{n=1}^N h(\mathbf{x}_n), \quad (1.4)$$

for $\mathbf{x}_n \sim f(\mathbf{x}) \forall n = 1 \dots N$ i.i.d. (“identically and independently distributed in f ”). One can immediately verify that

$$\left\langle \frac{1}{N} \sum_{n=1}^N h(\mathbf{x}_n) \right\rangle = \langle h \rangle_{\mathbf{x}}, \quad \text{Var} \left(\frac{1}{N} \sum_{n=1}^N h(\mathbf{x}_n) \right) = \frac{\text{Var}(h)}{N}$$

Exercise Prove the relations above.

Solution:

1. By linearity:

$$\left\langle \frac{1}{N} \sum_{n=1}^N h(\mathbf{x}_n) \right\rangle = \frac{1}{N} \sum_{n=1}^N \langle h(\mathbf{x}_n) \rangle = \frac{1}{N} \sum_{n=1}^N \langle h(\mathbf{x}) \rangle = \langle h \rangle_{\mathbf{x}}$$

2. For the properties of the variance:

$$\text{Var} \left(\frac{1}{N} \sum_{n=1}^N h(\mathbf{x}_n) \right) \stackrel{(*)}{=} \frac{1}{N^2} \sum_{n=1}^N \text{Var} h(\mathbf{x}_n) = \frac{N}{N^2} \text{Var} h(\mathbf{x}) = \frac{\text{Var}(h)}{N}$$

where in $(*)$ we used the fact that the covariance of independent variables is zero. ■

If instead we try to compute a d -dimensional integral using the Riemann sum approximation, we would have

$$\langle h \rangle_{\mathbf{x}} = \sum_{n=1}^N h(\mathbf{x}_n) f(\mathbf{x}_n) (\Delta x)^d + \mathcal{O}(N^{-1/d}),$$

where Δx is the linear spacing of the d -dimensional grid over the x domain. In this case, though, the bias has a too bad scaling with the dimension (i.e. exponential), because for high d most of the Lebesgue measure is “in the corners” of the domain (*curse of dimensionality*). Thus, this method is not recommended for volume estimation in high dimension.

To compute integrals of the kind of Eq. (1.3) we have to find a way to sample variables from a given distribution, which in our case is often a posterior

$$f(\boldsymbol{\theta} | \mathbf{y}_{\text{obs}}) = \frac{f(\mathbf{y}_{\text{obs}} | \boldsymbol{\theta}, \mathbf{x}) f(\boldsymbol{\theta})}{\int f(\mathbf{y}_{\text{obs}} | \boldsymbol{\theta}, \mathbf{x}) f(\boldsymbol{\theta}) d\boldsymbol{\theta}} \propto f(\mathbf{y}_{\text{obs}} | \boldsymbol{\theta}) f(\boldsymbol{\theta})$$

Since the normalization at the denominator is expensive, we’d like to avoid computing it. For now, we assume that the posterior density can be evaluated in reasonable time up to the normalization factor.

1.2.1 General building blocks of samplers

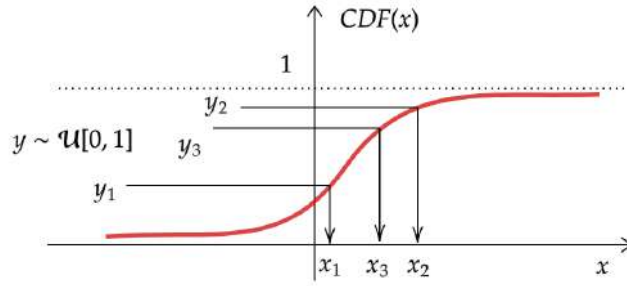
Here we propose some methods of drawing x from a given pdf $f(x)$.

Transformations

Suppose that we want to sample $x \sim f(x)$. We can always search for a change of variables $y = y(x)$ in which the new pdf is uniform in the interval $[0, 1]$, namely $f(x) dx = f(y) dy$ with $f(y) = \mathcal{U}[0, 1]$. Since $f(y)$ is uniform (constantly equal to unity in $[0, 1]$) we can remove it from the previous equation, and we get

$$f(x) = \frac{dy}{dx} \Rightarrow \underbrace{\int_{-\infty}^x f(x') dx'}_{\text{CDF}(x)} = \int_{-\infty}^x \frac{dy}{dx'} dx' = y(x)$$

This means that $x = \text{CDF}^{-1}(y)$ with $y \sim \mathcal{U}[0, 1]$, so if we’re able to compute the inverse cumulative density function, then we can generate samples according to $f(x)$ by simply generating random numbers uniformly in $[0, 1]$. Notice however that this method only works in 1D.

Figure 1.2: Sampling of $x \sim f(x)$ using the cumulative density function

Importance sampling

Instead of computing the normalization of $f(x)$, we can think to run a weighted average in which the weights are already normalized:

$$\langle h \rangle_x = \int h(x) f(x) dx = \int h(x) \frac{f(x)}{\tilde{f}(x)} \tilde{f}(x) dx \stackrel{MC}{\approx} \frac{1}{N} \sum_{i=1}^N h(x_i) \omega_i; \quad \omega_i = \frac{f(x_i)}{\tilde{f}(x_i)}; \quad x \sim \tilde{f}(x)$$

In practice, we firstly calculate the un-normalized weights $\hat{\omega}_i = f_{nn}(x_i)/\tilde{f}(x_i)$, where f_{nn} is an un-normalized version of f (e.g., if f is a posterior, f_{nn} could be the product of likelihood and prior), then we simply normalize the weights: $\omega_i = \hat{\omega}_i / \sum_j \hat{\omega}_j$. However, using this strategy we're only shifting the difficulty to finding a good approximation for the target function. This method is thus helpful if the new sampling function is sufficiently easy to draw from and if weights are reasonably close to 1 (i.e. if $\tilde{f}(x)$ is close to $f(x)$).

An easy example could be the following: suppose you want to generate random number according to some biased dice that you have in mind, but you have at disposal only fair dices. How can we simulate the biased sampling from the physical-unbiased dice? This is a typical example where importance sampling is used and, by looking at the previous equation, we can identify $f(x)$ as the pdf of the biased dice and $\tilde{f}(x)$ as the pdf of the unbiased one. Hence, using this method we are able to calculate averages over $f(x)$ by sampling x from $\tilde{f}(x)$.

Accept/reject

The accept/reject method is a classical sampling method which allows to sample from a distribution for which it is impossible to compute the inverse cumulative density function (CDF^{-1}). Instead, draws are taken from a uniform distribution and accepted with a proper probability. With reference to Figure 1.3, consider a suitable interval A which contains almost all the domain of x , and F as a number greater or equal to the maximum/supremum of the pdf. Then, the idea is to toss randomly points in the rectangle $A \times F$ and to retain only those points that fall under the curve. In practice we have to:

1. Generate $x \sim \mathcal{U}[A]$
2. Generate $r \sim \mathcal{U}[0, 1]$
3. Retain x only if $r < \frac{f(x)}{F} \equiv p_{acc}$

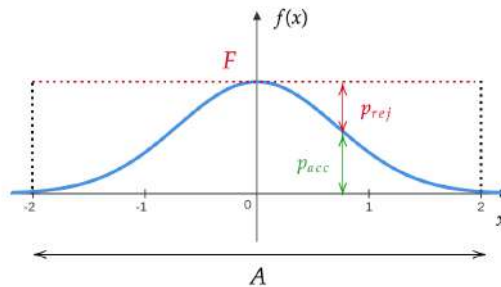


Figure 1.3: Accept/reject sampling scheme

Unfortunately, also this method suffers the curse of dimensionality, and in its pure form it becomes useless in high dimension problems. However, it is an important building block for more powerful sampling algorithms.



MCMC is a class of techniques for sampling from a probability distribution and can be used to estimate the distribution of parameters given a set of observations.

The central task of the whole Bayesian numerics is to get hold of the whole posterior distribution, either via approximation (i.e. Variational Bayes with ML) or via Monte Carlo sampling. From now on, we assume that we have a density $\pi(\boldsymbol{\theta})$ (e.g. a posterior), which can be evaluated in reasonable time ($10^4 - 10^6$ times) up to a normalization constant. We can do it as discussed in the previous chapter, with a simple accept/reject method, and at the end of the day we will run into issues caused by the curse of dimensionality (exponential loss of efficiency with the dimension of the sampling space). In this chapter, we will discuss instead a local search and design a stochastic process $P(\boldsymbol{\theta}'|\boldsymbol{\theta})$ which generates a Markov chain that is meant to sample the target distribution $\pi(\boldsymbol{\theta})$.

Definition 2.1 — Markov chain. A Markov chain is a stochastic process describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event. In other words, it doesn't have memory.

By doing such a sampling we will see that the efficiency will decay as a power-law of the dimension (d^{-1}). The price to pay is that the samples will be no longer independent but auto-correlated, leading to a reduced effective sample size. To pursue this approach, we must require $P(\boldsymbol{\theta}'|\boldsymbol{\theta})$ to have $\pi(\boldsymbol{\theta})$ as its equilibrium distribution, and this is directly related to have the **detailed balance** condition:

$$P(\boldsymbol{\theta}'|\boldsymbol{\theta})\pi(\boldsymbol{\theta}) = P(\boldsymbol{\theta}|\boldsymbol{\theta}')\pi(\boldsymbol{\theta}') \quad (2.1)$$

Definition 2.2 — Stationary Distributions of Markov Chains. A stationary distribution of a Markov chain is a probability distribution that remains unchanged in the Markov chain as time progresses. In other words, after an initial transient, the Markov process will eventually generate samples according to this stationary distribution.

In particular, the detailed balance condition implies that π is a stationary distribution of the Markov process under exams. If we denote with $\hat{\mathbf{P}}$ its propagator, then it acts on a probability distribution $f(\boldsymbol{\theta})$ as

$$(\hat{\mathbf{P}}f)(\boldsymbol{\theta}) := \int P(\boldsymbol{\theta}|\boldsymbol{\theta}') f(\boldsymbol{\theta}') d\boldsymbol{\theta}'$$

However, we will see that the stronger equilibrium condition (Eq. (2.1)) leads to a real-valued spectra which lead to favorable convergence properties. Also, we have that the stationary distribution is invariant under the operator $\hat{\mathbf{P}}$

$$(\hat{\mathbf{P}}\pi)(\boldsymbol{\theta}) = \int P(\boldsymbol{\theta}|\boldsymbol{\theta}') \pi(\boldsymbol{\theta}') d\boldsymbol{\theta}' \stackrel{(2.1)}{=} \int P(\boldsymbol{\theta}'|\boldsymbol{\theta}) \pi(\boldsymbol{\theta}) d\boldsymbol{\theta}' = \pi(\boldsymbol{\theta}) \int P(\boldsymbol{\theta}'|\boldsymbol{\theta}) d\boldsymbol{\theta}' \stackrel{(*)}{=} \pi(\boldsymbol{\theta})$$

where in (*) we use the fact that the probability to take a jump whatever is 1.

Definition 2.3 — Ergodicity. A system (either a dynamical system or a stochastic process) is said to be ergodic if any $\boldsymbol{\theta}' \in \Theta$ can be reached from any $\boldsymbol{\theta} \in \Theta$ with a positive probability and within a finite amount of jumps.

Proposition 2.1 If the process defined by $\hat{\mathbf{P}}$ satisfies the detailed balance condition with respect to π and is ergodic, then (under mild extra conditions) $(\hat{\mathbf{P}}^n f_0)(\boldsymbol{\theta}) \rightarrow \pi(\boldsymbol{\theta})$ in total variation for any initial probability density f_0 . More formally, for all subsets $A \subset \Theta$ of the parameters' domain we have that

$$\left| \int_A (\hat{\mathbf{P}}^n f_0)(\boldsymbol{\theta}) d\boldsymbol{\theta} - \int_A \pi(\boldsymbol{\theta}) d\boldsymbol{\theta} \right| = \mathcal{O}(e^{-rn}) \quad (2.2)$$

For some $r > 0$.

Proof. Let \mathbf{P} be an operator acting on a generic function f as follows:

$$(\mathbf{P}f)(\boldsymbol{\theta}) = \int f(\boldsymbol{\theta}') P(\boldsymbol{\theta}'|\boldsymbol{\theta}) d\boldsymbol{\theta}'$$

where we are integrating on the final point. Here, f belongs to the Hilbert space $\mathcal{H} = L^2_\pi(\boldsymbol{\theta})$ such that:

$$\langle f|g \rangle \equiv \int f(\boldsymbol{\theta}) g(\boldsymbol{\theta}) \pi(\boldsymbol{\theta}) d\boldsymbol{\theta}$$

This operator has two properties:

1. It is **self-adjoint**: $\mathbf{P} = \mathbf{P}^\dagger \implies \langle f|\mathbf{P}g \rangle = \langle \mathbf{P}f|g \rangle$;
2. It has unitary norm: $\|\mathbf{P}\| \equiv \sup_{\|f\|=1} \|\mathbf{P}f\| = 1$.

Exercise Prove the properties of \mathbf{P} operator in the proof of proposition 2.1.

Solution:

1. The self-adjointness is trivially proven by direct application:

$$\begin{aligned} \langle \mathbf{P}^\dagger f|g \rangle &= \langle f|\mathbf{P}g \rangle = \int f(\boldsymbol{\theta}) \left[\int g(\boldsymbol{\theta}') P(\boldsymbol{\theta}'|\boldsymbol{\theta}) d\boldsymbol{\theta}' \right] \pi(\boldsymbol{\theta}) d\boldsymbol{\theta} \\ &\stackrel{(*)}{=} \int \left[\int f(\boldsymbol{\theta}) P(\boldsymbol{\theta}|\boldsymbol{\theta}') d\boldsymbol{\theta} \right] g(\boldsymbol{\theta}') \pi(\boldsymbol{\theta}') d\boldsymbol{\theta}' = \langle \mathbf{P}f|g \rangle \end{aligned}$$

Where in the (*) step we exploited the detailed balance property (Eq. (2.1)).

2. The unitary norm can be proven by use of the Cauchy–Schwarz inequality for integrals:

$$\begin{aligned} \|\mathbf{P}f\|^2 &= \langle \mathbf{P}f|\mathbf{P}f \rangle = \int \pi(\boldsymbol{\theta}) (Pf)^2(\boldsymbol{\theta}) d\boldsymbol{\theta} \\ (Pf)^2(\boldsymbol{\theta}) &= \left(\int f(\boldsymbol{\theta}') P(\boldsymbol{\theta}'|\boldsymbol{\theta}) d\boldsymbol{\theta}' \right)^2 \leq \int f^2(\boldsymbol{\theta}') P(\boldsymbol{\theta}'|\boldsymbol{\theta}) d\boldsymbol{\theta}' \\ \implies \|\mathbf{P}f\|^2 &\leq \int f^2(\boldsymbol{\theta}') \underbrace{P(\boldsymbol{\theta}'|\boldsymbol{\theta}) \pi(\boldsymbol{\theta})}_{P(\boldsymbol{\theta}|\boldsymbol{\theta}') \pi(\boldsymbol{\theta}')} d\boldsymbol{\theta} d\boldsymbol{\theta}' = \|f\|^2 \end{aligned}$$

But since $\|\mathbf{P}\mathbb{1}\| = 1$ it follows $\|\mathbf{P}\| = 1$

As a consequence, the spectrum of the operator is real-valued (property 1) and belongs to the interval $[-1; 1]$ (property 2, since the spectrum is bounded by the operator's norm). One may also notice that $\lambda = 1$ is an eigenvalue of \mathbf{P} , in fact for $f = \mathbb{1}$ it holds:

$$(\mathbf{P}f)(\boldsymbol{\theta}) = \int f(\boldsymbol{\theta}') P(\boldsymbol{\theta}'|\boldsymbol{\theta}) d\boldsymbol{\theta}' = \int \mathbb{1} \cdot P(\boldsymbol{\theta}'|\boldsymbol{\theta}) d\boldsymbol{\theta}' = 1$$

since the marginalization of probability density function over the full domain of parameters returns 1. Furthermore, from the theory of ergodicity, an ergodic operator has a simple (unique) eigenvalue 1 and all other eigenvalues will have modulus smaller than 1 (this is a requirement to have an irreducible representation of the algebra generated by \mathbf{P}). Now, if $\varepsilon(\boldsymbol{\theta})$ is an eigenstate of \mathbf{P} relative to the eigenvalue λ , from the eigenvalues' equation we have

$$(\mathbf{P}\varepsilon)(\boldsymbol{\theta}) = \lambda \varepsilon(\boldsymbol{\theta}) \implies (\mathbf{P}^n \varepsilon)(\boldsymbol{\theta}) = \lambda^n \varepsilon(\boldsymbol{\theta})$$

This means that if we apply \mathbf{P} many times, in the end the only eigenspace that will survive is the one that belongs to $\mathbb{1}$, and the rest of the spectrum will shrink to zero because it is strictly smaller than 1. Therefore, under mild extra conditions there exist two (potentially asymmetric) **spectral gaps** such that \mathbf{P} has a spectrum $\sigma(\mathbf{P})$ like the one in the following picture.



Figure 2.1: Spectrum of the ergodic operator \mathbf{P} .

To make these statements more precise let us use a test function $g \in \mathcal{H}$ such that:

$$\langle g \rangle_{\pi} \equiv \int g(\boldsymbol{\theta}) \pi(\boldsymbol{\theta}) d\boldsymbol{\theta} = \langle \mathbb{1} | g \rangle = 1$$

Then:

$$\begin{aligned} \|\mathbf{P}^n g - \mathbb{1}\|^2 &= \langle \mathbf{P}^n g - \mathbb{1} | \mathbf{P}^n g - \mathbb{1} \rangle = \underbrace{\langle \mathbf{P}^n g | \mathbf{P}^n g \rangle}_{=\langle g | \mathbf{P}^{2n} g \rangle} - \underbrace{\langle \mathbb{1} | \mathbf{P}^n g \rangle}_{=(\mathbf{P}^n \mathbb{1} | g)} - \underbrace{\langle \mathbf{P}^n g | \mathbb{1} \rangle}_{=(g | \mathbf{P}^n \mathbb{1})} + \langle \mathbb{1} | \mathbb{1} \rangle = \\ &\stackrel{(*)}{=} \langle g | \mathbf{P}^{2n} g \rangle - 1 - 1 + 1 = \langle g | \mathbf{P}^{2n} g \rangle - 1 \end{aligned}$$

Where in (*) we exploited the fact that \mathbf{P}^n is self-adjoint and that the identity $\mathbb{1}$ is the eigenstate of \mathbf{P} with eigenvalue 1. Applying now the spectral theorem one can find the spectral representation of \mathbf{P} :

$$\mathbf{P} = \int_{\sigma(\mathbf{P})} \lambda dE_{\lambda} \implies \langle g | \mathbf{P} g \rangle = \int_{\sigma(\mathbf{P})} \lambda \langle g | dE_{\lambda} g \rangle \equiv \int_{\sigma(\mathbf{P})} \lambda d\mu_g(\lambda) \quad (2.3)$$

Where $d\mu_g(\lambda) \equiv \langle g | dE_{\lambda} g \rangle$ is the measure of the spectrum induced by the test function g on dE_{λ} by projection.

Moreover, for $\lambda = 1$ it holds $\mu_g(1) = \langle g | \mathbb{1} \rangle \langle \mathbb{1} | g \rangle = |\langle \mathbb{1} | g \rangle|^2 = 1$, by using g properties and the fact that the eigenstate of 1 is the identity. Applying now \mathbf{P} multiple times and combining with the previous results:

$$\begin{aligned} \langle g | \mathbf{P}^{2n} g \rangle &= \int_{\sigma(\mathbf{P})} \lambda^{2n} d\mu_g(\lambda) \implies \\ \|\mathbf{P}^n g - \mathbb{1}\|^2 &= \langle g | \mathbf{P}^{2n} g \rangle - 1 = \int_{\sigma(\mathbf{P})} \lambda^{2n} d\mu_g(\lambda) - 1 = \int_{\sigma(\mathbf{P}) \setminus \{1\}} \lambda^{2n} d\mu_g(\lambda) \leq |\lambda_{\max}|^{2n} \end{aligned}$$

Since the "-1" term cancels with the contribution to the spectral representation of eigenvalue "1". The term λ_{\max} in the equation above represents the eigenvalue of the remaining spectrum (red box in fig 2.1) with the

largest absolute value and given the fact that $\forall \lambda \quad |\lambda| \leq |\lambda_{\max}| < 1$ the last term goes exponentially to zero. So, in the end the norm can be upper bounded:

$$\|\mathbf{P}^n g - \mathbb{1}\|^2 \leq |\lambda_{\max}|^{2n} = \mathcal{O}(e^{-rn}) \text{ with } r = 2 \log(1/|\lambda_{\max}|)$$

Thus we observe that $\mathbf{P}^n g \rightarrow \mathbb{1}$.

To obtain the proposition's result is now sufficient to observe that:

$$\hat{\mathbf{P}}^n f = \pi \mathbf{P}^n \left(\frac{f}{\pi} \right)$$

From which the claim follows (for a generic test function the result can be obtained by re-scaling so to have a null-mean unitary-variance distribution). ■

Exercise Prove that $\hat{\mathbf{P}}^n f = \pi \mathbf{P}^n \left(\frac{f}{\pi} \right)$

Solution:

$$(\pi \mathbf{P}^n) \left[\frac{f}{\pi} \right] (\boldsymbol{\theta}) = \int \frac{f(\boldsymbol{\theta}_n)}{\pi(\boldsymbol{\theta}_n)} P(\boldsymbol{\theta}_n | \boldsymbol{\theta}_{n-1}) P(\boldsymbol{\theta}_{n-1} | \boldsymbol{\theta}_{n-2}) \dots P(\boldsymbol{\theta}_1 | \boldsymbol{\theta}) \pi(\boldsymbol{\theta}) d\boldsymbol{\theta}_1 \dots d\boldsymbol{\theta}_n$$

Using the detailed balance property iteratively we can write

$$\begin{aligned} & P(\boldsymbol{\theta}_n | \boldsymbol{\theta}_{n-1}) P(\boldsymbol{\theta}_{n-1} | \boldsymbol{\theta}_{n-2}) \dots P(\boldsymbol{\theta}_3 | \boldsymbol{\theta}_2) P(\boldsymbol{\theta}_2 | \boldsymbol{\theta}_1) \pi(\boldsymbol{\theta}_1) P(\boldsymbol{\theta} | \boldsymbol{\theta}_1) = \\ & P(\boldsymbol{\theta}_n | \boldsymbol{\theta}_{n-1}) P(\boldsymbol{\theta}_{n-1} | \boldsymbol{\theta}_{n-2}) \dots P(\boldsymbol{\theta}_3 | \boldsymbol{\theta}_2) \pi(\boldsymbol{\theta}_2) P(\boldsymbol{\theta}_1 | \boldsymbol{\theta}_2) P(\boldsymbol{\theta} | \boldsymbol{\theta}_1) = \\ & \vdots \\ & = \pi(\boldsymbol{\theta}_n) P(\boldsymbol{\theta} | \boldsymbol{\theta}_1) P(\boldsymbol{\theta}_1 | \boldsymbol{\theta}_2) \dots P(\boldsymbol{\theta}_{n-1} | \boldsymbol{\theta}_n) \end{aligned}$$

Hence

$$(\pi \mathbf{P}^n) \left[\frac{f}{\pi} \right] (\boldsymbol{\theta}) = \int \frac{f(\boldsymbol{\theta}_n)}{\pi(\boldsymbol{\theta}_n)} \cancel{\pi(\boldsymbol{\theta}_n)} P(\boldsymbol{\theta} | \boldsymbol{\theta}_1) P(\boldsymbol{\theta}_1 | \boldsymbol{\theta}_2) \dots P(\boldsymbol{\theta}_{n-1} | \boldsymbol{\theta}_n) d\boldsymbol{\theta}_1 \dots d\boldsymbol{\theta}_n = \left(\hat{\mathbf{P}}^n f \right) (\boldsymbol{\theta})$$

2.1.2 The problem of correlations

Definition 2.4 — Burn-in. Burn-in is a colloquial term that describes the practice of throwing away some iterations at the beginning of a Markov Chain Monte Carlo (MCMC) run. This can be done in order to enter a high probability region, a place where the states of the Markov chain are more representative of the distribution being sampled. In other words, we remove the first part of the chain mostly affected by the exponentially decreasing error $\exp(-nr)$ in the equations above.

The name of this practice comes from electronics. Many electronics components fail quickly (those which appertain to a less-reliable subset). So a burn-in is done at the factory to eliminate the worst ones.

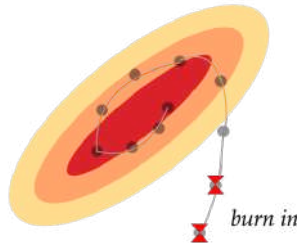


Figure 2.2: Burn-in example.

Once we have done this operation, we can use the MCMC (minus the burn-in) to estimate expectation values. As we've already observed, with MC methods we get rid of the bias, but we have the problem of

noise. In order to estimate the amount of noise, we define a test function g such that $\langle g \rangle = \langle 1|g \rangle = 0$ and variance $\sigma^2 = \langle g|g \rangle - (\langle g|1 \rangle)^2 = 1$. After we sample $\{\theta_1, \dots, \theta_N\}$ using MCMC, for large N we calculate the variance of the average of the values assumed by g as¹

$$\text{Var} \left(\frac{1}{N} \sum_{n=1}^N g(\theta_n) \right) = \frac{1}{N^2} \sum_{n,m} \langle g(\theta_n) g(\theta_m) \rangle \approx \frac{1}{N^2} \sum_{n \in \mathbb{Z}} \langle g(\theta_0) g(\theta_n) \rangle = \frac{1}{N} \left(1 + 2 \sum_{n=1}^{\infty} \langle g(\theta_0) g(\theta_n) \rangle \right)$$

Before analyzing where each term comes from, we want to clarify the notation used: $n = 0$ correspond to the diagonal terms of the matrix, $n = 1$ the first off-diagonal terms, $n = 2$ to the second ones and so on. Then, the "1" term comes from the variance $\langle g(\theta_0) g(\theta_0) \rangle$ and the "2" factor by considering only positive indexes. The approximation comes from the fact that, the larger n , i.e. the further we are from the diagonal, the fewer terms we have; but here we do as if we had N terms for each n . For N large, however, the error we make through this mis-counting is sub-leading. One may also notice that the first contribution to the variance above is brought by $1/N = \delta g/N$, which is the standard error of a Monte Carlo simulation.

We can see from this last formula that the variance has a non-null contribution depending on the **integrated auto-correlation function** τ_g and in particular the first term of τ_g is the variance of g (assumed to be unity in this case) divided by N , representing the typical error of a Monte Carlo process whose noise scales as σ/\sqrt{N} . **The total length of the Markov chain divided by τ_g quantifies the loss of samples ratio** due to auto-correlations and depends on the test function. This means that if we sample points through a MCMC method, only a fraction of them will be effectively useful to perform calculations, while the remaining part is not contributing because of the correlations. In the end we get an "effective" sample size that determines the amount of "useful" points given by the process.

Exercise — Prove that $\langle g(\theta_0) g(\theta_n) \rangle = \langle g | \mathbf{P}^n g \rangle$

Solution:

$$\langle g(\theta_0) g(\theta_n) \rangle = \int g(\theta_0) g(\theta_n) \pi(\theta_0, \theta_n) d\theta_0 d\theta_n$$

where

$$\pi(\theta_0, \theta_n) = \int P(\theta_n | \theta_{n-1}) P(\theta_{n-1} | \theta_{n-2}) \dots P(\theta_1 | \theta_0) \pi(\theta_0) d\theta_{n-1} \dots d\theta_1$$

hence

$$\begin{aligned} \langle g(\theta_0) g(\theta_n) \rangle &= \int g(\theta_0) g(\theta_n) P(\theta_n | \theta_{n-1}) P(\theta_{n-1} | \theta_{n-2}) \dots P(\theta_1 | \theta_0) \pi(\theta_0) d\theta_n \dots d\theta_0 \\ &= \langle g(\theta_0) | \int g(\theta_n) P(\theta_n | \theta_{n-1}) P(\theta_{n-1} | \theta_{n-2}) \dots P(\theta_1 | \theta_0) d\theta_n \dots d\theta_1 \rangle \\ &= \langle g | \mathbf{P}^n g \rangle \end{aligned}$$

With the result from the previous exercise and recalling Eq. (2.3), we find that τ_g can be written as

$$\tau_g = 1 + 2 \sum_{n=1}^{\infty} \langle g | \mathbf{P}^n g \rangle = 1 + 2 \sum_{n=1}^{\infty} \int_{\sigma(\mathbf{P})} \lambda^n d\mu_g(\lambda) = 1 + 2 \int_{\sigma(\mathbf{P})} \left(\sum_{n=1}^{\infty} \lambda^n \right) d\mu_g(\lambda)$$

$$\text{but } 1 = \langle g | g \rangle = \left\langle g \left| \underbrace{\int_{\sigma(\mathbf{P})} dE_{\lambda}}_{=1} g \right. \right\rangle = \int_{\sigma(\mathbf{P})} \langle g | dE_{\lambda} g \rangle = \int_{\sigma(\mathbf{P})} d\mu_g(\lambda)$$

¹

$$\begin{aligned} \text{Var} \left(\sum_{n=1}^N X_n \right) &= \sum_{n=1}^N \text{Var}(X_n) + 2 \sum_{m>n}^N \text{Cov}(X_n, X_m) = \sum_{n,m=1}^N \text{Cov}(X_n, X_m) = \sum_{n,m=1}^N \langle (X_n - \underbrace{\langle X_n \rangle}_{=0}) (X_m - \underbrace{\langle X_m \rangle}_{=0}) \rangle \\ &= \sum_{n,m=1}^N \langle X_n X_m \rangle \end{aligned}$$

$$\text{thus } \tau_g = \int_{\sigma^*(\mathbf{P})} \left(1 + 2 \frac{\lambda}{1 - \lambda}\right) d\mu_g(\lambda) = \int_{\sigma^*(\mathbf{P})} \frac{1 + \lambda}{1 - \lambda} d\mu_g(\lambda) \quad (2.4)$$

The geometric series $\sum_{n=1}^{\infty} \lambda^n$ converges only if $|\lambda| < 1$. We can recover this result since given the fact that $\langle g \rangle = \langle \mathbb{1} | g \rangle = 0$ then $\mu_g(1) = 0$ (measure of contribution to eigenspace of $\lambda = 1$ is null). The other eigenvalues, belonging to $\sigma^*(\mathbf{P}) = \sigma(\mathbf{P}) \setminus \{1\}$, are all smaller than one, since the spectrum is shrinking. One may easily notice that the integrand of the equation above can become arbitrarily large when $\lambda \rightarrow 1$. As a result, **the upper spectral gap is responsible for the amount of auto-correlations**. On the other hand, the convergence rate can be determined by either the upper or the lower (since it depends on the absolute value of λ). For example, in case of very large upper gap and small lower gap we can have very slow oscillating modes (i.e., the chain jumps from one side to the other of the distribution) that decay in long times. However, once they are gone and the process converges to stationarity they do not contribute to the auto-correlations anymore.

2.2 Metropolis algorithm

Metropolis is a specific implementation of MCMC. It is a method for sampling from a random distribution (target distribution), even if we don't know the normalization constant. To do this, we construct and sample from a Markov Chain whose stationary distribution is the target distribution we are looking for. Algorithms of this type are generally used for sampling from multi-dimensional distributions, especially when the number of dimensions is high.

Algorithm 2.1 — Metropolis

The procedure consists of two steps that are repeated for a large number of iterations:

1. Make a proposal of a new θ' , drawing it from a jump (a **proposal**) distribution $t(\theta'|\theta)$;
2. Accept the proposal with an **acceptance probability**^a $a(\theta'|\theta) = \min\left(1, \frac{\pi(\theta')}{\pi(\theta)}\right)$.

★

^a A slight generalization of the Metropolis algorithm is the Metropolis-Hastings algorithm, which allows for non-symmetric jump distributions. The Metropolis accept/reject probability then has to be replaced by $a(\theta'|\theta) = \min\left(1, \frac{\pi(\theta') t(\theta|\theta')}{\pi(\theta) t(\theta'|\theta)}\right)$.

In this last operation, if $a(\theta'|\theta) = 1$ then for sure we will make the chain jump from the previous to the proposed candidate. If not, $0 \leq a(\theta'|\theta) < 1$ takes the value of that ratio and so we will still accept the candidate with a probability given by the ratio itself. In the first case the step is advantageous, while in the second we still can move to the new parameter drawn, but only with probability $a(\theta'|\theta)$ (this acts as a correction). The delicate choice is to use a suitable $t(\theta'|\theta)$.

It is important to stress that the sampling is done **on the posterior** probability distribution, because we want to evaluate the probability over the parameters space. Indeed the ratio is done in order to evaluate if the new proposal is more probable respect the actual state; moreover we can exploit the ratio as

$$\frac{\pi(\theta')}{\pi(\theta)} = \frac{p(\mathbf{x}|\theta')p(\theta')}{\int p(\mathbf{x}|\theta')p(\theta')d\theta'} \frac{\int p(\mathbf{x}|\theta)p(\theta)d\theta}{p(\mathbf{x}|\theta)p(\theta)}$$

where we can see that the evidence terms cancel out.

To put this into formulae, we write

$$P(\theta'|\theta) = a(\theta'|\theta)t(\theta'|\theta) + \delta(\theta - \theta') \overbrace{\left(1 - \int a(\theta'|\theta)t(\theta'|\theta)d\theta'\right)}^{r(\theta)}$$

Exercise Prove that this probability, with an acceptance probability $a(\theta'|\theta)$ and a symmetric proposal jump distribution, satisfies the detailed balance condition. Prove that ergodicity is satisfied (for a Gaussian t).

Solution:

We need to prove that $P(\boldsymbol{\theta}'|\boldsymbol{\theta})\pi(\boldsymbol{\theta}) = P(\boldsymbol{\theta}|\boldsymbol{\theta}')\pi(\boldsymbol{\theta}')$. So we start from the definition

$$a(\boldsymbol{\theta}'|\boldsymbol{\theta})t(\boldsymbol{\theta}'|\boldsymbol{\theta})\pi(\boldsymbol{\theta}) + \delta(\boldsymbol{\theta} - \boldsymbol{\theta}')\pi(\boldsymbol{\theta})r(\boldsymbol{\theta}) = a(\boldsymbol{\theta}|\boldsymbol{\theta}')t(\boldsymbol{\theta}|\boldsymbol{\theta}')\pi(\boldsymbol{\theta}') + \delta(\boldsymbol{\theta} - \boldsymbol{\theta}')\pi(\boldsymbol{\theta}')r(\boldsymbol{\theta}')$$

The delta sets to zero the rejection part since $\boldsymbol{\theta}' \neq \boldsymbol{\theta}$ and the proposal jump distribution is symmetric: $t(\boldsymbol{\theta}'|\boldsymbol{\theta}) = t(\boldsymbol{\theta}|\boldsymbol{\theta}')$, so we are left with

$$a(\boldsymbol{\theta}'|\boldsymbol{\theta})\pi(\boldsymbol{\theta}) = a(\boldsymbol{\theta}|\boldsymbol{\theta}')\pi(\boldsymbol{\theta}') \quad (2.5)$$

And to show this identity we go by cases

- $\pi(\boldsymbol{\theta}) > \pi(\boldsymbol{\theta}')$, so $a(\boldsymbol{\theta}'|\boldsymbol{\theta}) = 1$ and we can invert Eq. 2.5

$$a(\boldsymbol{\theta}|\boldsymbol{\theta}') = \frac{\pi(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta}')}$$

that is equal to the definition of $a(\boldsymbol{\theta}|\boldsymbol{\theta}')$ as the acceptance rate.

- $\pi(\boldsymbol{\theta}') > \pi(\boldsymbol{\theta})$, so $a(\boldsymbol{\theta}|\boldsymbol{\theta}') = 1$ and we can invert Eq. 2.5

$$a(\boldsymbol{\theta}'|\boldsymbol{\theta}) = \frac{\pi(\boldsymbol{\theta}')}{\pi(\boldsymbol{\theta})}$$

that is equal to the definition of $a(\boldsymbol{\theta}'|\boldsymbol{\theta})$ as the acceptance rate. ■

To find out how the efficiency (convergence and auto-correlation) scales with d we now determine the spectral gap. We firstly define the Metropolis operator $P = M + K$ with

$$(Mf)(\boldsymbol{\theta}) = f(\boldsymbol{\theta}) \left(1 - \int a(\boldsymbol{\theta}'|\boldsymbol{\theta})t(\boldsymbol{\theta}'|\boldsymbol{\theta})d\boldsymbol{\theta}' \right)$$

$$(Kf)(\boldsymbol{\theta}) = \int f(\boldsymbol{\theta}')a(\boldsymbol{\theta}'|\boldsymbol{\theta})t(\boldsymbol{\theta}'|\boldsymbol{\theta})d\boldsymbol{\theta}'$$

where the first is the **multiplication** operator, which essentially represents the **rejection probability** (probability for which we do not jump) $r(\boldsymbol{\theta})$, and its spectrum (which is continuous and bounded by 0 and 1 since M is positive and upper-limited) is simply given by the essential range of the rejection probability $r(\boldsymbol{\theta})$ which, for non-pathological target functions t , is bounded away from unity and thus it exists a spectral gap. The **essential range** of a function is intuitively the “non-negligible” range of the function: it does not change between two functions that are equal almost everywhere. One way of thinking of the essential range of a function is the set on which the range of the function is most “concentrated”. More precisely, the value of λ determining the spectrum of M and its upper spectral gap is given by

$$\sup \left\{ \lambda \in \mathbb{R} \mid \int \chi(|r(\boldsymbol{\theta}) - \lambda| < \varepsilon) \pi(\boldsymbol{\theta}) d\boldsymbol{\theta} > 0, \forall \varepsilon > 0 \right\}$$

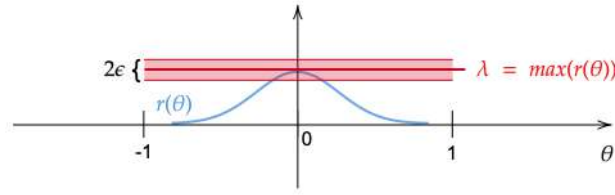
Looking more deeply into the definition, we observe that $\int \chi \pi(\boldsymbol{\theta}) d\boldsymbol{\theta} \equiv \langle \chi \rangle_\pi$, and recalling that the expectation value of the indicator function χ represents the probability of its argument to be true, we can rewrite the previous definition as follows:

$$\sup \{ \lambda \in \mathbb{R} \mid P(|r(\boldsymbol{\theta}) - \lambda| < \varepsilon) > 0, \forall \varepsilon > 0 \}$$

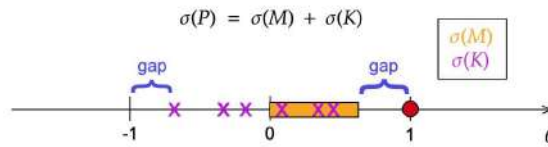
In particular, as one may see from Figure 2.3, for a continuous rejection probability $r(\boldsymbol{\theta})$ the delimiter of the essential range is $\lambda = \max(r(\boldsymbol{\theta}))$. The previous definition, however, applies also to a wider range of less-regular distributions.

The second term of P is the **kernel** operator K , which is in particular a Hilbert-Schmidt operator, and therefore a compact operator if

$$\|K\|^2 = \text{Tr}\{K^\dagger K\} = \int \min\{\pi(\boldsymbol{\theta}), \pi(\boldsymbol{\theta}')\} t(\boldsymbol{\theta}'|\boldsymbol{\theta})^2 d\boldsymbol{\theta} d\boldsymbol{\theta}' < \infty$$

Figure 2.3: Example of essential range limit detection for a continuous $r(\theta)$

Compact operators have the nice property to have a discrete spectrum, so K will have it. According to Weyl's perturbation theorem, P has then the same essential spectrum (total spectrum minus the isolated eigenvalues) as M . That is, only a finite number of isolated eigenvalues can populate the spectral gaps left by M , and therefore also P has a spectral gap. The **spectral gap** can either be determined by the continuous spectrum of M or the discrete one of K : this depends on the size, σ , of the jump distribution.

Figure 2.4: Spectrum of ergodic operator P from M 's and K 's spectra. Since M is positive definite, will have a spectrum only in $[0, +1]$.

In fact, looking at the kernel operator we have at Taylor's 1st order in the small jump size limit:

$$\begin{aligned}
 t(\theta'|\theta) &\simeq \delta(\theta - \theta') + \mathcal{O}(\sigma^2) \implies \\
 (Kf)(\theta) &= \int f(\theta') a(\theta'|\theta) t(\theta'|\theta) d\theta' \simeq f(\theta) + \mathcal{O}(\sigma^2) \equiv \lambda f(\theta) \implies \\
 \varepsilon_{gap} &\equiv 1 - \lambda_{max} \simeq \mathcal{O}(\sigma^2)
 \end{aligned} \tag{2.6}$$

If $t(\theta'|\theta)$ is large (big jumps), then there is a large rejection probability $r(\theta)$ and thus $\lambda_{max} \in \sigma(M)$ (gap determined by M); on the contrary, little jumps assure a little rejection probability and thus the essential spectrum will be small: as a consequence, there will likely be sporadic points belonging to the spectrum of K that populate the intervals outside $\sigma(M)$ and therefore $\lambda_{max} \in \sigma(K)$ (gap determined by K).

Scaling with dimension

Let us see how the supremum of the spectrum of P scales with the dimension d : let us choose $t(\theta'|\theta)$ as a Gaussian with covariance matrix $\sigma^2 \cdot \mathbb{1}$. It turns out that when d becomes large, σ^2 needs to become small, otherwise the rejection probability $r(\theta)$ grows too much.

Indeed, the reasoning leading to Eq. (2.6) can be applied also to the rejection probability: at Taylor's 1st order it holds

$$r(\theta) = 1 - \int a(\theta'|\theta) t(\theta'|\theta) d\theta' \simeq \mathcal{O}(\sigma^2 \cdot d)$$

Thus, to have $r(\theta)$ bounded away from 1 it is necessary that at least $\sigma^2 \propto d^{-1}$. However, from Eq. (2.6) we see that we neither want σ^2 to scale faster than d^{-1} , otherwise the discrete spectrum of K becomes too much close to 1. At the end of the day this means that the best scaling for the jump size is

$$\sigma^2 \propto \frac{1}{d} \propto \varepsilon_{gap} \tag{2.7}$$

In this way, also the gap decreases linearly while enlarging the dimensionality of the parameters' domain. As a consequence, the fraction of rejected point in sampling with MCMC Metropolis is exponentially smaller than using other global sampling methods; in this case, once convergence is reached, the only source of losses is due to the auto-correlations. From Eq. (2.4) we see that $\tau_g \sim \varepsilon_{gap}^{-1} \sim d$, meaning that these losses increase linearly with the dimension and not exponentially like in the brute-force sampling algorithms reported above.

In a nutshell: All the math done above shows us that there is a trade-off in the tuning of the jump size σ : we don't want to jump too far, i.e. large σ , otherwise we lose efficiency due to a low acceptance rate (continuous part of the spectrum close to one) and neither to make jumps too small, otherwise it takes time to explore the whole space, which in spectral terms it means that we have discrete eigenvalues that appear close to 1 and so we have a low number of effective samples.

2.3 Metropolis algorithm tuning

Metropolis acceptance-rejection grants that, for any jump distribution $t(\boldsymbol{\theta}'|\boldsymbol{\theta})$ we choose, we will converge to our target distribution $\pi(\boldsymbol{\theta})$. Lesson 3 29/03

We consider as jump distribution $t(\boldsymbol{\theta}'|\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$, where Σ is the covariance matrix, that in the simplest case can be chosen as $\Sigma = \sigma^2 \cdot \mathbb{1}$ (meaning that is just a diagonal matrix and we jump with size σ in all directions). We have then to take into account the following trade-off: GC FC

1. A too small σ will lead to a long exploration time of the parameters range;
2. A too large σ will lead to over-jump and so to a low acceptance rate.

This trade-off, as explained in the previous section, can be visualized in the trend of the spectrum.

Now, looking a bit more into this tuning procedure and sticking to $t(\boldsymbol{\theta}'|\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$, we focus on a more general covariance matrix Σ .

Ideally, we would like to use a covariance matrix that resembles the shape of the target distribution, namely $\Sigma \propto \text{Cov}(\boldsymbol{\pi})$ (see figure 2.5 for an intuitive picture). This is however difficult, because it requires the tuning of $\frac{d(d+1)}{2} \sim \mathcal{O}(d^2)$ parameters, and in higher dimensions this becomes intractable. Luckily us, there is a class of algorithms that does this tuning automatically, namely **adaptive Metropolis** class. Two examples are reported in the paragraph below.

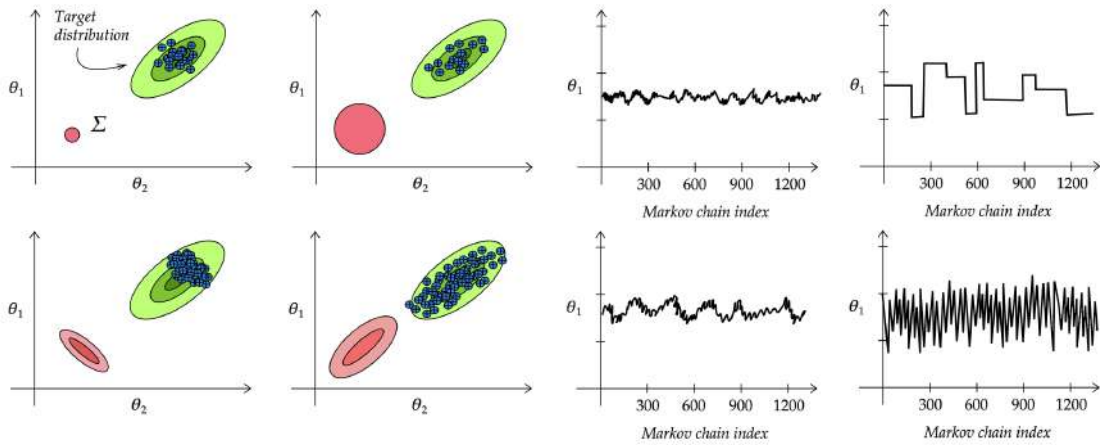


Figure 2.5: Top-left) with a too small fixed jumping size it's difficult for the Markov chain to explore the lowest probability area of the distribution and moreover it will takes a longer time for the whole distribution to get sampled.

Top-right) a too high fixed jumping size is not likely to be accepted: the process gets stuck for many steps.

Bottom-left) now the jumps are taken from a probability distribution. However, if the shape of the jumping distribution doesn't fit properly the target distribution, we end up sampling only a small region of the target.

Bottom-right) with the correct tuning of the jump distribution we are able to sample correctly the target.

2.3.1 Haario algorithm

This algorithm [HST01], starting from the assumption that the MC is sampling the target distribution $\pi(\boldsymbol{\theta})$, simply uses the history of the chain to estimate $\text{Cov}(\boldsymbol{\pi})$. So, at step $n + 1$, Σ is chosen to be proportional to the **empirical covariance** $\hat{\text{Cov}}$ of the history:

$$\Sigma_{n+1} = \beta \hat{\text{Cov}} + \varepsilon \mathbb{1} \quad (2.8)$$

where

$$\hat{\text{Cov}} = \frac{1}{n-1} \sum_{i=1}^n (\boldsymbol{\theta}_i - \bar{\boldsymbol{\theta}})(\boldsymbol{\theta}_i - \bar{\boldsymbol{\theta}})^T, \quad \bar{\boldsymbol{\theta}} = \frac{1}{n} \sum_{i=1}^n \boldsymbol{\theta}_i$$

in which β is a tuning parameter and $\varepsilon \mathbb{1}$ prevents the process from degenerating into some linear sub-space of the parameter space and get stuck on it, that would lead to lose ergodicity.

Exercise Why is there a $n-1$ and not an n in front of the sum? Prove that given $\boldsymbol{\theta}_i \sim \text{i.i.d.}$ random variables $i = 1, \dots, n$, then

$$\left\langle \frac{1}{n-1} \sum_{i=1}^n (\boldsymbol{\theta}_i - \bar{\boldsymbol{\theta}})(\boldsymbol{\theta}_i - \bar{\boldsymbol{\theta}})^T \right\rangle$$

is an unbiased estimator of $\text{Cov}(\boldsymbol{\theta})$.

Solution:

By definition

$$\text{Cov}(\boldsymbol{\theta}) \equiv \langle (\boldsymbol{\theta} - \langle \boldsymbol{\theta} \rangle)(\boldsymbol{\theta} - \langle \boldsymbol{\theta} \rangle)^T \rangle = \langle \boldsymbol{\theta} \boldsymbol{\theta}^T \rangle - \langle \boldsymbol{\theta} \rangle \langle \boldsymbol{\theta}^T \rangle$$

while we have

$$\mathbb{E}[\hat{\text{Cov}}(\boldsymbol{\theta})] = \left\langle \frac{1}{n-1} \sum_{i=1}^n (\boldsymbol{\theta}_i - \bar{\boldsymbol{\theta}})(\boldsymbol{\theta}_i - \bar{\boldsymbol{\theta}})^T \right\rangle = \frac{1}{n-1} \sum_{i=1}^n \left(\langle \boldsymbol{\theta}_i \boldsymbol{\theta}_i^T \rangle - \langle \boldsymbol{\theta}_i \bar{\boldsymbol{\theta}}^T \rangle - \langle \bar{\boldsymbol{\theta}} \boldsymbol{\theta}_i^T \rangle + \langle \bar{\boldsymbol{\theta}} \bar{\boldsymbol{\theta}}^T \rangle \right) \quad (2.9)$$

In general, since the variables are i.i.d., we have

$$\langle \boldsymbol{\theta}_i \boldsymbol{\theta}_j^T \rangle = \begin{cases} \langle \boldsymbol{\theta}_i \boldsymbol{\theta}_i^T \rangle = \langle \boldsymbol{\theta} \boldsymbol{\theta}^T \rangle & \text{if } i = j \\ \langle \boldsymbol{\theta}_i \boldsymbol{\theta}_j^T \rangle \stackrel{\text{i.i.d.}}{=} \langle \boldsymbol{\theta}_i \rangle \langle \boldsymbol{\theta}_j^T \rangle = \langle \boldsymbol{\theta} \rangle \langle \boldsymbol{\theta}^T \rangle & \text{if } i \neq j \end{cases}$$

Expanding the various terms:

$$\langle \boldsymbol{\theta}_i \bar{\boldsymbol{\theta}}^T \rangle = \langle \bar{\boldsymbol{\theta}} \boldsymbol{\theta}_i^T \rangle = \frac{1}{n} \left(\langle \boldsymbol{\theta}_i \boldsymbol{\theta}_i^T \rangle + \sum_{i \neq j} \langle \boldsymbol{\theta}_i \boldsymbol{\theta}_j^T \rangle \right) = \frac{1}{n} \langle \boldsymbol{\theta} \boldsymbol{\theta}^T \rangle + \frac{n-1}{n} \langle \boldsymbol{\theta} \rangle \langle \boldsymbol{\theta}^T \rangle$$

$$\begin{aligned} \langle \bar{\boldsymbol{\theta}} \bar{\boldsymbol{\theta}}^T \rangle &= \left\langle \left(\frac{1}{n} \sum_{i=1}^n \boldsymbol{\theta}_i \right) \left(\frac{1}{n} \sum_{j=1}^n \boldsymbol{\theta}_j \right)^T \right\rangle = \frac{1}{n^2} \left[\sum_{i=1}^n \langle \boldsymbol{\theta}_i \boldsymbol{\theta}_i^T \rangle + \sum_{i > j} \left(\langle \boldsymbol{\theta}_i \rangle \langle \boldsymbol{\theta}_j^T \rangle + \langle \boldsymbol{\theta}_j \rangle \langle \boldsymbol{\theta}_i^T \rangle \right) \right] \\ &= \frac{1}{n^2} \left[n \langle \boldsymbol{\theta} \boldsymbol{\theta}^T \rangle + n(n-1) \langle \boldsymbol{\theta} \rangle \langle \boldsymbol{\theta}^T \rangle \right] = \frac{1}{n} \langle \boldsymbol{\theta} \boldsymbol{\theta}^T \rangle + \frac{n-1}{n} \langle \boldsymbol{\theta} \rangle \langle \boldsymbol{\theta}^T \rangle \end{aligned}$$

Inserting in Eq. (2.9) gives

$$\begin{aligned} \mathbb{E}[\hat{\text{Cov}}(\boldsymbol{\theta})] &= \frac{1}{n-1} \sum_{i=1}^n \left[\langle \boldsymbol{\theta} \boldsymbol{\theta}^T \rangle - \frac{2}{n} \langle \boldsymbol{\theta} \boldsymbol{\theta}^T \rangle - 2 \frac{n-1}{n} \langle \boldsymbol{\theta} \rangle \langle \boldsymbol{\theta}^T \rangle + \frac{1}{n} \langle \boldsymbol{\theta} \boldsymbol{\theta}^T \rangle + \frac{n-1}{n} \langle \boldsymbol{\theta} \rangle \langle \boldsymbol{\theta}^T \rangle \right] \\ &= \frac{n}{n-1} \left[\frac{n-1}{n} \langle \boldsymbol{\theta} \boldsymbol{\theta}^T \rangle - \frac{n-1}{n} \langle \boldsymbol{\theta} \rangle \langle \boldsymbol{\theta}^T \rangle \right] = \langle \boldsymbol{\theta} \boldsymbol{\theta}^T \rangle - \langle \boldsymbol{\theta} \rangle \langle \boldsymbol{\theta}^T \rangle = \text{Cov}(\boldsymbol{\theta}) \end{aligned}$$

To get a good estimation of this empirical covariance, a rich enough history, i.e. a large number of accepted samples, is needed. Then, this algorithm is often combined with the **delayed rejection** technique. When a proposal, e.g. $t_1(\boldsymbol{\theta}_1 | \boldsymbol{\theta}_0)$, gets rejected, we try to profit from the gained information and propose a second, shorter (could be simply just half as far), jump in the same direction. This second jump is thus depending on two positions: $t_2(\boldsymbol{\theta}_2 | \boldsymbol{\theta}_1, \boldsymbol{\theta}_0)$. With such jumps, we need to adapt the Metropolis acceptance-rejection probability in order to maintain detailed balance

$$\alpha_1(\boldsymbol{\theta}_1, \boldsymbol{\theta}_0) = \min \left(1, \frac{\pi(\boldsymbol{\theta}_1) t_1(\boldsymbol{\theta}_0 | \boldsymbol{\theta}_1)}{\pi(\boldsymbol{\theta}_0) t_1(\boldsymbol{\theta}_1 | \boldsymbol{\theta}_0)} \right) \quad (2.10)$$

Supposing that this first jump has been rejected because α_1 was rather small, we now attempt the second one

with t_2 which has to be accepted with a modified acceptance probability α_2 such as:

$$\alpha_2(\boldsymbol{\theta}_2, \boldsymbol{\theta}_1, \boldsymbol{\theta}_0) = \min \left(1, \frac{\overbrace{\pi(\boldsymbol{\theta}_2) t_2(\boldsymbol{\theta}_1 | \boldsymbol{\theta}_2) t_1(\boldsymbol{\theta}_0 | \boldsymbol{\theta}_1, \boldsymbol{\theta}_2)}^{\boldsymbol{\theta}_2 \rightarrow \boldsymbol{\theta}_1 \rightarrow \boldsymbol{\theta}_0} (1 - \alpha_1(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2))}{\overbrace{\pi(\boldsymbol{\theta}_0) t_2(\boldsymbol{\theta}_1 | \boldsymbol{\theta}_0) t_1(\boldsymbol{\theta}_2 | \boldsymbol{\theta}_1, \boldsymbol{\theta}_0)}^{\boldsymbol{\theta}_2 \leftarrow \boldsymbol{\theta}_1 \leftarrow \boldsymbol{\theta}_0} (1 - \alpha_1(\boldsymbol{\theta}_1, \boldsymbol{\theta}_0))} \right) \quad (2.11)$$

Where the terms $(1 - \alpha_1(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2))$ and $(1 - \alpha_1(\boldsymbol{\theta}_1, \boldsymbol{\theta}_0))$ take into account the probability that the first jump has been rejected.

Exercise Show that detailed balance is maintained

2.3.2 Vihola algorithm

An algorithm that also tries to learn the covariance structure of the target on the flight is the Vihola algorithm [Vih12]. This one attempts to enforce a user-defined acceptance rate α^* through re-scaling, depending on what we have just learned, of the jump covariance matrix Σ_n in the direction of the previous jump. So, every

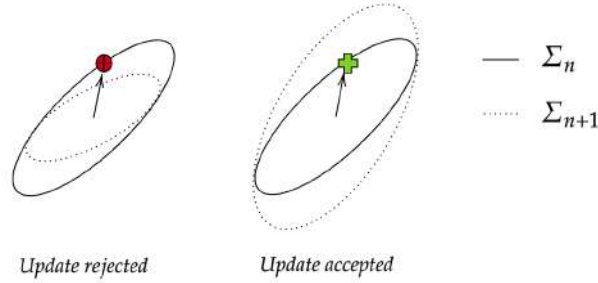


Figure 2.6: Possible covariance rescalings according to the Vihola algorithm.

time we jump, we try to squeeze and inflate the covariance structure along the tested direction, depending on the α value retrieved by the previous jump (if the α obtained it's far too small we have probably over-jumped, so next time we will decrease the jump, squeezing Σ in this direction. On the other hand, if α is too close to 1, we probably need to inflate Σ in that direction, through a larger jump).

In order to mathematically implement this, Vihola uses the *Cholesky decomposition* of the covariance matrix Σ_n :

$$\Sigma_n = L_n L_n^T$$

in which L_n is a lower triangular matrix with strictly positive diagonal entries. Given $\mathbf{u} = (u_1, \dots, u_n)^T$, where each $u_i \sim \mathcal{N}(0, 1)$ are standard normal samples, we simply have to apply L to it to obtain a sample with the desired covariance $L\mathbf{u} \sim \mathcal{N}(0, \Sigma)$. This is very easy to prove:

$$\begin{aligned} \langle L\mathbf{u} \rangle &= L \langle \mathbf{u} \rangle \\ \langle (L\mathbf{u})(L\mathbf{u})^T \rangle &= L \langle \mathbf{u}\mathbf{u}^T \rangle L^T \stackrel{(*)}{=} LL^T = \Sigma \end{aligned}$$

where in $(*)$ we used that $\langle \mathbf{u}\mathbf{u}^T \rangle = \mathbb{1}$ is the covariance of the standard normal distribution.

Algorithm 2.2 — Vihola

The Vihola algorithm iterates the 3 following steps:

1. at step n of the procedure, sample a vector of i.i.d. standard normal variables $\mathbf{u}_n \sim \mathcal{N}(0, 1)$ and make a jump proposal:

$$\boldsymbol{\theta}_n^* = \boldsymbol{\theta}_{n-1} + L_{n-1} \mathbf{u}_n$$

2. accept-reject (same as vanilla Metropolis) with probability^a

$$\alpha_n = \min \left(1, \frac{\pi(\boldsymbol{\theta}_n^*)}{\pi(\boldsymbol{\theta}_{n-1})} \right)$$

3. update Σ_n along the direction of the attempted jump according to

$$\Sigma_n = L_n L_n^T = L_{n-1} \left(\mathbb{1} + \eta_n (\alpha_n - \alpha^*) \frac{\mathbf{u}_n \mathbf{u}_n^T}{\|\mathbf{u}_n\|^2} \right) L_{n-1}^T,$$

where η_n is the *learning rate* belonging to a predefined diminishing sequence and α^* is the desired mean acceptance rate. In the expression above we recognize in brackets the projection matrix onto the 1D subspace spanned by \mathbf{u}_n , which is multiplied by the term $(\alpha_n - \alpha^*)$ that keeps Σ constant once we reach the desired acceptance probability. ★

^asince we have a symmetric jump distribution we can stick to this simple version

2.3.3 EMCEE sampler

These adaptive Metropolis algorithm try to learn the covariance structure on the fly, but if we have a target distribution that has a non-linear correlation structure, e.g. it is “banana-shaped”, its correlation structure will depend on the parameters range considered. This means that it is not possible to find a one-size-fits-all jump distribution, and the structure learned will be a compromise between the two branches. In such situations,

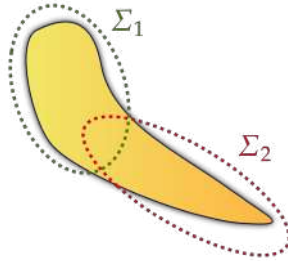


Figure 2.7: “Banana” distribution on which we identify two regions associated to different covariances.

interacting particle methods allow to overcome this issue to some extent: the idea is to evolve several Markov chains (instead of a single one) that all together form a population and let them learn from each other by interaction. We start from a parameters’ space in which we have different starting points, also called *particles*. While iterating, we randomly choose one among them and we make a jump in the direction identified by a randomly chosen partner θ'' (not necessarily in the starting set of particles). If the jump is accepted, then we are going to delete the starting particles, otherwise we leave it there. Such algorithms can also harness the power of parallel computing infrastructure. An example of this idea is the EMCEE-sampler, implementing the *stretch move*.

Algorithm 2.3 — EMCEE stretch move

In one step within the iterations:

1. we randomly select a θ in the set of initial particles;
2. we randomly select a θ'' in the initial space of particles ^a and we identify a vector between the two. The proposal θ' will be a point along the stretched or loosed (by a scaling factor z) vector direction.

$$\theta' = \theta'' + \mathcal{Z} (\theta - \theta''),$$

in which $\mathcal{Z} \sim f(z)$ is a random variable, satisfying the following scaling condition $f(\frac{1}{z}) = z f(z)$; one can prove that $f(z) \sim \frac{1}{\sqrt{z}}$, usually with $z \in [\frac{1}{a}, a]$, in which a is a tuning parameter;

3. the proposal θ' is then accepted with a modified Metropolis probability

$$a(\theta' | \theta) = \min \left(1, z^{d-1} \frac{\pi(\theta')}{\pi(\theta)} \right) \quad (2.12)$$

^abe careful that in a general particle algorithm it is not required to choose one companion among the starting set

Exercise Prove that (2.12) satisfies detailed balance

Solution:

We want to prove that

$$P(\boldsymbol{\theta}'|\boldsymbol{\theta})\pi(\boldsymbol{\theta}) = P(\boldsymbol{\theta}|\boldsymbol{\theta}')\pi(\boldsymbol{\theta}') \quad (2.13)$$

If we move to polar coordinates centered at $\boldsymbol{\theta}''$, then

$$\begin{aligned} \pi(\boldsymbol{\theta})d\boldsymbol{\theta} &\propto \pi(r)r^{d-1}dr \\ \pi(\boldsymbol{\theta}')d\boldsymbol{\theta}' &\propto \pi(r')r'^{d-1}dr' \end{aligned}$$

and, since $z = \frac{r'}{r}$,

$$f(z)dz = f(r')dr' \implies f(r') = f(z)\frac{dz}{dr'} = f\left(\frac{r'}{r}\right)\frac{1}{r}$$

At the same time:

$$f(z)dz = f(r)dr \implies f(r) = f(z)\frac{dz}{dr} \stackrel{*}{=} f\left(\frac{r}{r'}\right)\frac{1}{r'}$$

Conditioning (2.13) on $\boldsymbol{\theta}''$, we get

$$\begin{aligned} P(\boldsymbol{\theta}'|\boldsymbol{\theta}, \boldsymbol{\theta}'')\pi(\boldsymbol{\theta}) &= f\left(\frac{r'}{r}\right) \min\left(1, \left(\frac{r'}{r}\right)^{d-1} \frac{\pi(r')}{\pi(r)}\right) \pi(r)r^{d-2} \\ &= f\left(\frac{r'}{r}\right) \min\left(\pi(r)r^{d-1}, \pi(r')r'^{d-1}\right) \frac{1}{r} \\ &\stackrel{*}{=} f\left(\frac{r}{r'}\right) (r')^{-1} \min\left(\pi(r')r'^{d-1}, \pi(r)r^{d-1}\right) \\ &= f\left(\frac{r}{r'}\right) \min\left(1, \left(\frac{r}{r'}\right)^{d-1} \frac{\pi(r)}{\pi(r')}\right) \pi(r')r'^{d-2} \\ &= P(\boldsymbol{\theta}|\boldsymbol{\theta}', \boldsymbol{\theta}'')\pi(\boldsymbol{\theta}') \end{aligned}$$

where in (*) we used the property $f(1/z) = zf(z)$

The strength of this algorithm is that the direction of the jump is **position dependent**, a characteristic that is not present in the standard Metropolis algorithm, which sets the same direction for every starting point. That's why stretch move methods work well for non-linear and also **multi-modal** target.

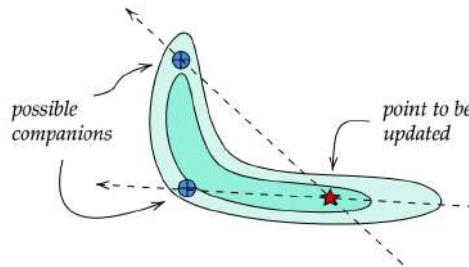


Figure 2.8: EMCEE stretch move. The direction of the updates depends on the structure of the distribution.

Convergence

Adaptive Monte Carlo algorithms (Haario and Vihola) are not Markov chains anymore, since they exploit their history, losing their Markovian-property: the convergence proofs become a lot harder, to the point that for the Vihola algorithm a general convergence proof can't be found. This problem can be bypassed by

using an adaptive algorithm just at the beginning and then, after the covariance structure is assumed to be learned, switching to the traditional Metropolis. On the other hand, the convergence proof of the interacting particle algorithm is rather easy, since here the jumps only depends on the current state of all the particles (it is therefore a Markov process).

2.4 Gibbs Sampling

Lesson 4 Gibbs sampling refers to the idea of sampling one component $x_i \in \mathbf{x}$ at a time, while keeping all the other components fixed (see Fig. 2.9). This strategy becomes very useful in high dimensional spaces when combined with the Metropolis algorithm, allowing a much higher acceptance rate that we wouldn't have by jumping toward random directions in the parameters' space.

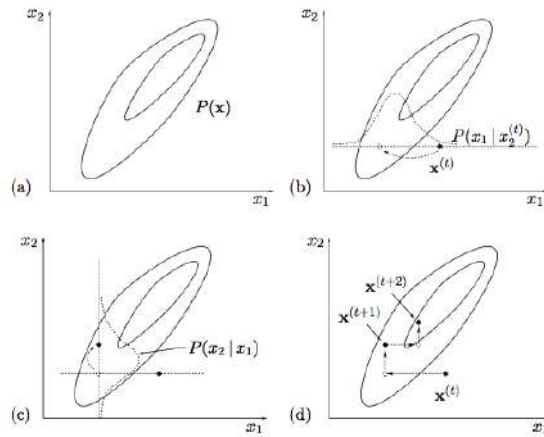


Figure 2.9: Visual explanation of Gibbs sampling.

Let's make an example to make things more concrete. Suppose to measure some quantity that gives in output N i.i.d. variables identified as $y_i \sim \mathcal{N}(\mu, \sigma)$, $i = 1, \dots, N$ (you can think, for example, to measure the heights of students in a class). A handy re-parameterization of the problem is by means of the so called *precision*, defined as $\tau = \sigma^{-2}$, so that the parameters that we want to infer are now $\theta = (\mu, \tau)$. The likelihood of this model, obtained as the product of the N $\mathcal{N}(y_i | \mu, \sigma)$ distributions, is

$$f(\mathbf{y} | \mu, \tau) \propto \tau^{\frac{N}{2}} \exp \left\{ -\frac{\tau}{2} \sum_{i=1}^N (y_i - \mu)^2 \right\},$$

while for the prior we can choose a so-called *conjugate prior*, meaning that both prior and posterior belongs to the same pdf family (or class). For example, if we choose a normal prior for the parameter μ

$$f(\mu) \propto \frac{1}{\sigma^*} \exp \left\{ -\frac{(\mu - \mu^*)^2}{2(\sigma^*)^2} \right\}$$

that is parametrized by the hyper-parameter $\theta = (\mu^*, \sigma^*)$, if we multiply it by a normal likelihood, the posterior is still a normal distribution (the product of gaussian pdfs is gaussian itself). This means that if we keep updating the prior with the likelihood, we stay in the same family of pdfs. This is particularly convenient, because this way we can visualize the learning as a curve in the space of hyper-parameters (μ^*, σ^*) . Instead, for the precision we shall use a gamma distribution that is again a conjugate prior, because if we multiply it by the likelihood we increase the parameters α, β but we keep having a gamma pdf.

$$f(\tau) = \Gamma(\tau; \alpha, \beta) \propto \tau^{\alpha-1} e^{-\beta\tau} \quad \alpha, \beta > 0$$

One of the major caveat in Bayesian analysis is that **we can't express knowing nothing**, because we need to propose a prior. However, we can try to introduce as less information as possible in the process by means of a conjugate prior distribution within the class of **non-informative priors** (sometimes called also *uninformative priors*), obtained as results of some proper limits of the parameters

$$f(\mu) \xrightarrow{\sigma^* \rightarrow \infty} 1$$

$$f(\tau) \xrightarrow{\alpha, \beta \rightarrow 0} \tau^{-1}$$

This way the final posterior is

$$f(\mu, \tau | \mathbf{y}_{obs}) \propto \tau^{\frac{N}{2}-1} \exp \left\{ -\frac{\tau}{2} \sum_{i=1}^N (y_i - \mu)^2 \right\}$$

Now, in order to perform a MCMC that samples (μ, σ) from this posterior, it is convenient to exploit the Gibbs sampling method. Indeed, if we keep τ fixed then we have that $f(\mu, \tau | \mathbf{y}_{obs})$ is a normal pdf, and if we keep μ fixed then we have a gamma pdf. In both cases it's easy to sample numbers from such distributions at the computational level. In summary, the Gibbs sampling method consists on the iteration of the following two steps:

1. Fix μ , extract $\tau \sim \Gamma\left(\frac{N}{2}, \frac{1}{2} \sum_{i=1}^N (y_i - \mu)^2\right)$
2. Fix τ , extract $\mu \sim \mathcal{N}\left(\frac{1}{N} \sum_{i=1}^N y_i, \frac{1}{N\tau}\right)$

However, in general we cannot perform direct sampling from the posterior. In these cases it's very useful to combine Gibbs sampling with the Metropolis algorithm, especially in high dimensional spaces. Recall that in the Metropolis algorithm one has to scale the size of the jump distribution as $\sigma^2 \sim d^{-1}$ in order to stay away from an exponential loss in the sampling. Nevertheless, if we now apply the Gibbs sampling, just a fraction of the parameters is changed at a time, and the rest is kept fixed. This means that we can make larger jumps, at the cost of performing them in a subspace of the parameter's space.

■ **Example 2.1 — Ising model.** A typical physical example where this procedure can be used successfully is the Ising model, where we want to sample Ising configurations, i.e. classical spins $x_i = \pm 1$, on a lattice. These configurations are distributed as

$$f(\mathbf{x} | \beta, h) = \mathcal{Z}^{-1}(\beta, h) e^{-\beta H(\mathbf{x}, h)}$$

where the Hamiltonian is

$$H(\mathbf{x}, h) = - \sum_{\langle i, j \rangle} x_i x_j + h \sum_i x_i$$

and the first summation is performed over nearest neighbors. Here the Gibbs sampling procedure consists on keeping fixed all spins but one, which has to be flipped, obtaining a new configuration $\mathbf{x} \rightarrow \mathbf{x}'$. Then, the usual Metropolis accept/reject step is performed with acceptance ratio

$$a(\mathbf{x}' | \mathbf{x}) = \min \left(1, e^{-\beta [H(\mathbf{x}', h) - H(\mathbf{x}, h)]} \right) = \min \left(1, e^{-\beta \Delta H} \right) \quad (2.14)$$

and we iterate until thermalization. Note that here we are not doing Bayesian inference, but we're just simulating the system's configurations from the likelihood given some fixed parameters (β, h) . The opposite problem, i.e. given the configuration we want to infer (β, h) , is instead a much harder problem. This is due to the fact that sampling from the posterior $f(\beta, h | \mathbf{x}_{obs})$ with the classical Metropolis requires to compute the likelihood many times, but since it contains the partition function this would be very expensive. For this kind of tasks a more affordable approach is given by the Approximate Bayesian Computation (see next chapter). ■

■ **Example 2.2 — Rainfall-runoff modelling.** As mentioned in the first chapter, traditionally if people want to predict the runoff of a river based on rain forecast, they use models of the kind

$$\mathbf{y}_{runoff}(\mathbf{r}, \boldsymbol{\theta}) = \mathbf{y}_{det}(\mathbf{r}, \boldsymbol{\theta}) + E(\mathbf{r}, \boldsymbol{\theta}),$$

where \mathbf{r} is the rain given as an input and $\boldsymbol{\theta}$ are the parameters of the model. However, it is known that one of the major sources of uncertainty comes from the rain measurements, and therefore it would be better to model the uncertainties directly on the input data instead of trying to describe them through the error modelling E . To tackle this problem, a possible strategy is to perform the model calibration by splitting into two parts the observations $\mathbf{y}_{obs} = (\mathbf{y}_{rain}, \mathbf{y}_{runoff})$, and to treat the real rain realizations \mathbf{x} as an unknown that

we'll try to infer together with the parameters of the model: $\mathbf{q} = (\mathbf{x}, \boldsymbol{\theta})$. This way we can write the joint model likelihood $f(\mathbf{y}_{rain}, \mathbf{y}_{runoff} | \boldsymbol{\theta})$ as the marginalization over **the unknown real rain** "hidden variable" \mathbf{x}

$$f(\mathbf{y}_{rain}, \mathbf{y}_{runoff} | \boldsymbol{\theta}) = \int f(\mathbf{y}_{runoff} | \mathbf{x}, \boldsymbol{\theta}) f(\mathbf{y}_{rain} | \mathbf{x}, \boldsymbol{\theta}) f(\mathbf{x} | \boldsymbol{\theta}) d\mathbf{x},$$

where $f(\mathbf{x} | \boldsymbol{\theta})$ is called "rain generator" because it generates rain time series (we will refer to them also as *trajectories* or *realizations*) according to some stochastic models and parameters $\boldsymbol{\theta}$. The result is that we moved from an intractable likelihood in a low dimensional space Θ to a relatively easy likelihood in a high dimensional space $\mathcal{Q} = (\Theta, X)$ with a nasty integral over all the rain realizations.

We shall therefore sample the \mathbf{q} 's from the joint posterior

$$f(\mathbf{x}, \boldsymbol{\theta} | \mathbf{y}_{obs}) \propto \underbrace{f(\mathbf{y}_{obs} | \mathbf{x}, \boldsymbol{\theta})}_{\text{likelihood}} \underbrace{f(\mathbf{x} | \boldsymbol{\theta}) f(\boldsymbol{\theta})}_{\text{prior}}$$

and consider them as microscopic states of some statistical mechanics problem

$$f(\mathbf{q} | \mathbf{y}_{obs}) \propto \mathcal{Z}^{-1}(\mathbf{y}_{obs}) e^{-H(\mathbf{q}, \mathbf{y}_{obs})}$$

At this point we can apply Gibbs-sampling together with the classical Metropolis algorithm as described above:

1. Fix a rain realization \mathbf{x} and sample $\boldsymbol{\theta}$ with Metropolis.
2. Fix $\boldsymbol{\theta}$ and sample a rain realization. This can also be done in a Gibbs-like way, meaning that we can select just a small time window, generate a piece of trajectory and update the full time series if the Metropolis accept/reject step allows it (see Fig. 2.10).

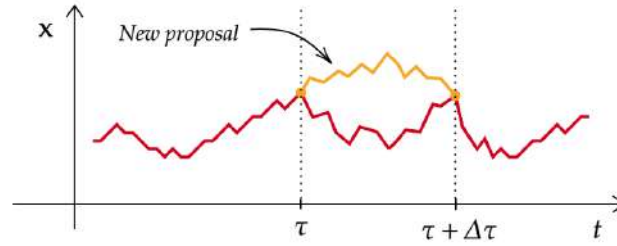


Figure 2.10: Updating a rain time series with Gibbs sampling. A realization is sampled between time τ and $\tau + \Delta\tau$, and the proposal can be accepted or not according to the Metropolis rule.

■

3. Hamiltonian Monte Carlo

3.1 Basic Concepts

So far we've seen how Metropolis algorithms can be used for sampling the posterior distribution of an inference problem. However, many times it happens that the strategy of performing random jumps in the parameters' space can be very inefficient, because many proposal are not accepted. For difficult inference problems it becomes of paramount importance that the algorithm converges to the target distribution in an acceptable time, and we therefore have to find out a strategy to increase the acceptance rate. Metropolis algorithms become a lot more efficient if we use the local shape of the posterior (i.e. its derivatives) to decide where to jump, and here is where Hamiltonian Monte Carlo (HMC) algorithms come in play. The idea is to introduce auxiliary *momentum variables* \mathbf{p} , for each degrees of freedom $\mathbf{q} = (\mathbf{x}, \boldsymbol{\theta})$, so that we can write the posterior as

$$f(\mathbf{q}|\mathbf{y}_{obs}) = \mathcal{Z}^{-1}(\mathbf{y}_{obs}) e^{-V(\mathbf{q}, \mathbf{y}_{obs})} \overset{(*)}{\propto} \mathcal{Z}^{-1}(\mathbf{y}_{obs}) \int \exp \left\{ -V(\mathbf{q}, \mathbf{y}_{obs}) - \sum_i \frac{p_i^2}{2m_i} \right\} d\mathbf{p}$$

where we've defined the potential $V(\mathbf{q}, \mathbf{y}_{obs}) = -\ln(f(\mathbf{y}_{obs}|\mathbf{q})f(\mathbf{q}))$ and where in $(*)$ we've introduced an identity up to a constant factor due to the integration of the Gaussian kinetic term. We can now define the Hamiltonian as

$$H(\mathbf{q}, \mathbf{p}; \mathbf{y}_{obs}) = T + V = \sum_i \frac{p_i^2}{2m_i} + V(\mathbf{q}, \mathbf{y}_{obs}) \quad (3.1)$$

so that the posterior pdf can be written as

$$f(\mathbf{q}|\mathbf{y}_{obs}) \propto \mathcal{Z}^{-1}(\mathbf{y}_{obs}) \int e^{-H(\mathbf{q}, \mathbf{p}; \mathbf{y}_{obs})} d\mathbf{p}$$

The previous equation now suggests to use Hamiltonian dynamics to retrieve a Markov chain with an high acceptance probability, that means: we no longer perform random jumps, but we rather follow the directions given by the Hamilton equations associated to the Hamiltonian (3.1). We know in fact that the Hamilton equations preserve the energy, so when we compute the acceptance probability (see Eq. (2.14)) we have an energy difference $\Delta E = 0$ that leads to an acceptance probability of 1. So, in principle, we will always accept the proposal update in the accept/reject step. This is actually quite not true, since we need to perform a discretization of the Hamilton equations that will unavoidably introduce some errors, invalidating the energy conservation. To mitigate this drawback we thus still keep a final accept-reject step. The intuition behind HMC is that we kick all the degrees of freedom with a random force (sampled from the Gaussian kinetic energy) and then let the system explore the associated energy-shell of the phase space. Sampling a new set of momenta in each iteration makes sure that all the energy shells are sampled.

In a nutshell: We've seen that the energy conservation property of the Hamiltonian dynamics allows to have an acceptance probability close to 1. However, how can we visualize the fact that the Hamiltonian somehow is able to describe the geometry of the target distribution?

Let's have a look at the picture in Fig.3.1. To sample from a probability distribution, it's trivial to say that we want to toss more points in the high probability region, an less points elsewhere. Thus, an efficient sampler should explore more likely the regions where our posterior has its maximum values. In HMC, what we do is to “encode” the negative logarithm of the un-normalized posterior in the potential of the Hamiltonian; this way, if we propagate the equations of motion for a certain duration τ , the dynamics will spontaneously explore the region around the minimum of the potential, corresponding to the high probability region of the posterior.^a

^aYou can find some nice animations at https://arogozhnikov.github.io/2016/12/19/markov_chain_monte_carlo.html

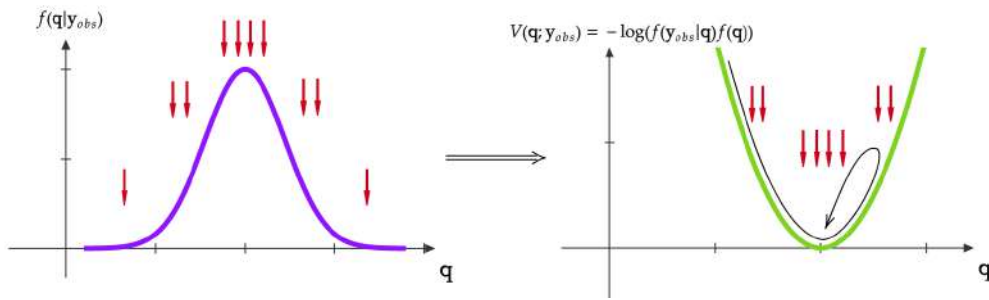


Figure 3.1: In HMC, we turn the posterior pdf into a potential and we exploit the Hamiltonian dynamics to explore the high probability region

As we mentioned above, to **simulate numerically** the Hamilton equations we are forced to introduce a discretization. In general, it is known that discretizations introduce errors in energy conservation, and we can handle them, but we can't allow errors in reversibility. Indeed, we have to choose a discretization that is **reversible in time** and **volume preserving**, meaning that the Liouville's theorem must hold (needed in order to guarantee the detailed balance condition).

An example of a discretization that preserves detailed balance is the **Leapfrog** scheme:

Algorithm 3.1 — Leap Frog

1. Initialize the momenta with some explicit methods like the Euler's one:

$$p_i\left(\tau + \frac{\Delta\tau}{2}\right) = p_i(\tau) - \frac{\partial V(q)}{\partial q_i}\Big|_{\tau} \Delta\tau + O(\Delta\tau^2)$$

2. Alternatively update positions and momenta

$$\begin{cases} q_i(\tau + \Delta\tau) = q_i(\tau) + p_i(\tau + \frac{\Delta\tau}{2}) \frac{\Delta\tau}{m_i} + O(\Delta\tau^3) \\ p_i(\tau + \frac{\Delta\tau}{2}) = p_i(\tau - \frac{\Delta\tau}{2}) - \frac{\partial V(q)}{\partial q_i}\Big|_{\tau} \Delta\tau + O(\Delta\tau^3) \end{cases} \quad (3.2)$$

★

Exercise Prove that the leap frog discretization is time reversible.

Solution: First, we can re-arrange the formula above and put the changed sign inside the time interval

$$\begin{cases} q_i(\tau) = q_i(\tau + \Delta\tau) + p_i(\tau + \frac{\Delta\tau}{2}) \left(-\frac{\Delta\tau}{m_i}\right) \\ p_i(\tau - \frac{\Delta\tau}{2}) = p_i(\tau + \frac{\Delta\tau}{2}) - \frac{\partial V(q)}{\partial q_i}\Big|_{\tau} (-\Delta\tau) \end{cases}$$

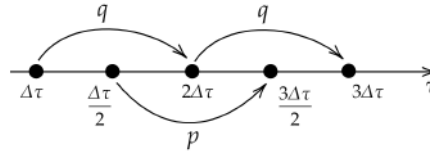


Figure 3.2: Leapfrog procedure.

Then, we can define a new time interval $\Delta\tau' = -\Delta\tau$, which is just the time reverse of the previous one; so we end up with:

$$\begin{cases} q_i(\tau) = q_i(\tau - \Delta\tau') + p_i(\tau - \frac{\Delta\tau'}{2}) \frac{\Delta\tau'}{m_i} \\ p_i(\tau + \frac{\Delta\tau'}{2}) = p_i(\tau - \frac{\Delta\tau'}{2}) - \frac{\partial V(q)}{\partial q_i} \Big|_{\tau} \Delta\tau' \end{cases}$$

The p 's equation is in the same form of the original Leapfrog scheme, while the q 's equation is not. The reason is that, when we invert the arrow of the time in a $q - p$ scheme, the first variable to be updated is p , and not q anymore (look at figure 3.2). In order to fix this we can just add a $\Delta\tau'$ term everywhere inside the parenthesis of the q , and finally we retrieve the original leapfrog scheme but in a reverse-time fashion. ■

Finally, in order to correct errors of order $O(\Delta\tau^2)$ in energy conservation, we need an accept/reject step of the Metropolis algorithm.

Algorithm 3.2 — HMC

Tuning parameters: $\{m_i\}_{i=1\dots N}$, τ ;

1. Sample momenta $\mathbf{p} \sim \exp\left\{-\sum_i \frac{p_i^2}{2m_i}\right\}$;
2. Propagate for a duration τ the Hamilton equations

$$\dot{q}_i = \frac{\partial H(\mathbf{q}, \mathbf{p})}{\partial p_i}, \quad \dot{p}_i = -\frac{\partial H(\mathbf{q}, \mathbf{p})}{\partial q_i}$$

using the leapfrog discretization;

3. Accept the proposal move with acceptance probability

$$\min\left(1, \exp\left\{-H(q(\tau + n\Delta\tau), p(\tau + n\Delta\tau)) + H(q(\tau), p(\tau))\right\}\right) \quad (3.3)$$

★

HMC methods are incredibly powerful, but the cost to pay is that we need to calculate derivatives (see Eq. (3.1)). Nowadays there exist highly-optimized routines as *Automated Differentiation* that uses meta-programming, meaning that they take a function in input and give you back another function which calculates the exact derivative of the first one.

3.2 Advanced Hamiltonian Monte Carlo

3.2.1 Example: Stochastic differential equation (SDE) model

To improve this basic scheme of Hamiltonian Monte Carlo we've seen so far, we shall go through an important example coming from hydrological models. This gives us the opportunity to introduce the stochastic differential equation models (in short, SDE), which are becoming quite popular in many fields. Consider a state variable $x(t)$ whose evolution is given by the *Langevin equation*

Lesson 5
07/04
FC
LR

$$\dot{x}(t) = r(t) - \rho(t)x(t) \quad (3.4)$$

For example, if we're describing the rainfall-runoff of a river by means of a bucket-model, then $x(t)$ can be the amount of water in a catchment, $r(t)$ is a (known) input variable representing the rain and $\rho(t)x(t)$ is the outflow (see Fig. 3.3). Reasonably, the outflow of the model is proportional to the amount of water in the tank, but we also introduce a variable $\rho(t)$ which represents a stochastic process. For this model we choose a

simple **stationary gaussian process**, meaning that it's completely described by only two time-independent momenta,

$$\rho_0 = \langle \rho(t) \rangle, \quad \langle (\rho(t) - \rho_0)(\rho(t') - \rho_0) \rangle = \delta(t - t') \rho_0 \gamma,$$

where the delta function means that the noise is totally uncorrelated. Often this term is also called *white noise*, because if we take the Fourier transform of the correlation function we find a constant value representing an equal contribution of all frequencies (hence the name *white*). The parameters here introduced are the mean of $\rho(t)$, ρ_0 ($[\rho_0] = t^{-1}$), and the dimensionless noise magnitude, γ .¹

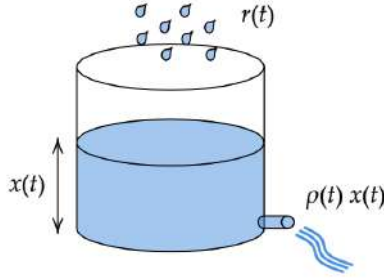


Figure 3.3: The model we use describes the evolution of the tank level $x(t)$ as a function of the incoming rain $r(t)$ and a stochastic term $\rho(t)$

In order to work with our model we also choose a discretization framework, that in our case will be **Itô discretization**

$$x_i = x(t_i), \quad t_{i+1} - t_i = \Delta t, \quad x_{i+1} = x_i + \Delta t(r_i - \rho_i x_i).$$

In the last formula, the term x_i in brackets could have been also $(x_{i+1} - x_i)/2$ (Stratonovich discretization) or other kind of discretizations, but this could lead, in principle, to an entirely different model. For this specific case, instead, the change of discretization would translate to just a change of the parametrization.

Given the discretization framework, we can rewrite ρ as

$$\rho_i = \rho_0 + \eta_i, \quad \langle \eta_i \eta_j \rangle = \gamma \rho_0 \frac{\delta_{ij}}{\Delta t} \longrightarrow \langle \eta_i \rangle \sim \sqrt{\langle \eta_i^2 \rangle} = \sqrt{\frac{\gamma \rho_0}{\Delta t}}$$

and we can finally get the discretized Langevin equation

$$x_{i+1} = x_i + \underbrace{\Delta t(r_i - \rho_0 x_i)}_{\text{Drift term}} + \underbrace{\sqrt{\Delta t \rho_0 \gamma} x_i \varepsilon_i}_{\text{Noise}}, \quad \varepsilon_i \sim \mathcal{N}(0, 1) \quad (3.5)$$

from which we can easily simulate realizations for the model. Notice that for small Δt the stochastic term becomes the dominant one, since it depends on $\sqrt{\Delta t}$, and from this observation we conclude that $\Delta x_i = x_{i+1} - x_i \sim \mathcal{O}(\sqrt{\Delta t})$. As a consequence, in the limit $\Delta t \rightarrow 0$ typical realizations of SDE will be continuous, but not differentiable anywhere. This also allows us to explain how changing the discretization can afflict the resulting model; in fact, if we imagine to choose $x_{i+1} = x_i + \Delta x$ in place of x_i , in the end we would get an additional term

$$x_{i+1} = x_i + \Delta t(r_i - \rho_i x_{i+1}) = x_i + \Delta t(r_i - \rho_i x_i) + \Delta t \Delta x (\rho_0 + \eta_i)$$

where $\Delta t \Delta x \rho_0 \sim \Delta t^{\frac{3}{2}}$ and so we will neglect it, while $\Delta t \Delta x \eta_i \sim \Delta t$ so we have

$$x_{i+1} = x_i + \Delta t(r_i - \rho_i x_i + c_i) = x_i + \Delta t(\tilde{r}_i - \rho_i x_i)$$

For this particular problem the effective difference would be only a reparametrization ($\tilde{r}_i \equiv r_i + c_i$), but in some cases it can lead to new terms and different behaviours.

We shall study now the special case for which we have a time-independent input $r(t) \equiv r_0$ and solve the model by finding the stationary distribution for x in the limit $t \rightarrow \infty$. By taking the average of Eq. (3.5)

¹It is worth to notice that the combination of variables chosen for right hand side of the correlation function is consistent in terms of dimensional analysis (recall that $[\delta(t - t')] = t^{-1}$)

we immediately find that the resulting distribution will have $\langle x \rangle = r_0/\rho_0$. Also, if in Eq. (3.4) we consider $x \gg r_0/\rho_0$ then the first term becomes negligible, and we get a scale-invariant equation²: $\dot{x} = r - \rho x \approx -\rho x$. Since power laws are the only scale-invariant distributions, our result must be power law-like in the right tail. By solving the model one can actually find that the final distribution is a inverse gamma distribution that satisfies the power law tail condition.

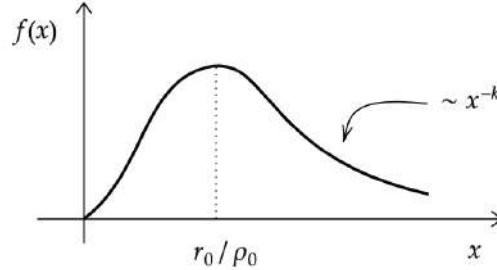


Figure 3.4: Resulting stationary distribution for x

What we described so far can be taken as a reference model, but what follows is true in general. Suppose we made a set of measurements $\{y_I\}$ at times $\{t_I\}$, $I = 1 \dots n$, such that $t_{I+1} - t_I \gg \Delta t$, meaning that there's a non trivial dynamics happening between two observation points. Our measurements will be subject to some amount of error that we suppose to be gaussian

$$y_I = x(t_I) + \sigma \tilde{\epsilon}_I, \quad \tilde{\epsilon}_I \sim \mathcal{N}(0, 1).$$

Our target is therefore to infer $\theta = (\rho_0, \gamma, \sigma)$. Often, this kind of problems are solved using a *Kalman filter*, which however require our states to be normally distributed. Instead, the model we've shown above is an example of a system in which the variables are distributed according to a non normal distribution with power-law tails, therefore this strategy wouldn't work in our case. A good way to proceed is instead using HMC. The posterior of the discretized model is of the form

$$f(\theta | \mathbf{y}_{obs}) = \int f(\mathbf{y}_{obs} | \mathbf{x}, \theta) f(\mathbf{x} | \theta) f(\theta) d\mathbf{x},$$

where $f(\mathbf{x} | \theta)$ is the component of the prior which is given by the stochastic model.

It is convenient to treat the realizations \mathbf{x} together with θ as the parameters of the model to be inferred, and to write the integral in the usual Boltzmann-like form. Finally, we can couple the parameters with the corresponding momenta $(\theta, \mathbf{x}) \leftrightarrow (\boldsymbol{\pi}, \mathbf{p})$ by introducing a kinetic term

$$\begin{aligned} f(\theta | \mathbf{y}_{obs}) &= \mathcal{Z}^{-1}(\mathbf{y}_{obs}) \int \exp[-V_{obs}(\mathbf{x}, \theta; \mathbf{y}_{obs}) - V_{model}(\mathbf{x}, \theta) - V_{prior}(\theta)] d\mathbf{x} \\ &= \mathcal{Z}^{-1}(\mathbf{y}_{obs}) \int \exp[-V_{tot}(\mathbf{x}, \theta; \mathbf{y}_{obs}) - K(\boldsymbol{\pi}, \mathbf{p})] d\mathbf{x} d\boldsymbol{\pi} d\mathbf{p} \\ &= \mathcal{Z}^{-1}(\mathbf{y}_{obs}) \int \exp[-H(\mathbf{x}, \theta, \boldsymbol{\pi}, \mathbf{p}; \mathbf{y}_{obs})] d\mathbf{x} d\boldsymbol{\pi} d\mathbf{p} \end{aligned}$$

where

$$H(\mathbf{x}, \theta, \boldsymbol{\pi}, \mathbf{p}; \mathbf{y}_{obs}) = V_{obs}(\mathbf{x}, \theta; \mathbf{y}_{obs}) + V_{model}(\mathbf{x}, \theta) + V_{prior}(\theta) + \sum \frac{p_i^2}{2m_i} + \sum \frac{\pi_\alpha^2}{2m_\alpha} \quad (3.6)$$

Note that we could take the formal limit $\Delta t \rightarrow 0$ and write this integral as a *path integral* over continuous realizations \mathbf{x} . A nice visualization of the problem is given by looking at the time series of \mathbf{x} as it was a swinging polymer chain constrained to obey some external potentials (see figure 3.5). In this case, V_{obs} forces the polymer to agree with the observations we made at times $\{t_I\}$, while V_{model} guarantees that the points between consecutive observations are a plausible realization of the model. In the same way the θ 's will explore the parameter's space according to our potentials.

We shall now look closely at the different components of the Hamiltonian:

²The model is invariant under a multiplication of x by some constant c

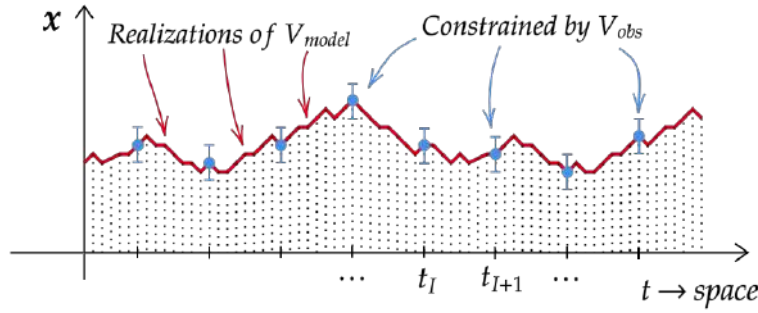


Figure 3.5: Interpretation of the realizations \mathbf{x} as a polymer constrained by V_{obs} in correspondence of the observations, and by V_{model} elsewhere

- $V_{prior}(\boldsymbol{\theta})$: typically is chosen as a simple function of $\boldsymbol{\theta}$, such as a log-normal distribution;
- $V_{obs}(\mathbf{x}, \boldsymbol{\theta}; \mathbf{y}_{obs})$: assuming to have observations with a symmetric, normal uncertainty, we can use the logarithm of a normal distribution

$$V_{obs}(\mathbf{x}, \boldsymbol{\theta}; \mathbf{y}_{obs}) = \frac{1}{2\sigma^2} \sum_{l=1}^n (x(t_l) - y_{obs,l})^2 + n \ln(\sqrt{2\pi}\sigma)$$

- $V_{model}(\mathbf{x}, \boldsymbol{\theta})$: this term has to be derived from the model. From Eq. (3.5) we can write the discretized Langevin equation

$$\Delta x_i = (r_i - \rho_0 x_i) \Delta t + \sqrt{\rho_0 \gamma \Delta t} x_i \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, 1)$$

and therefore the associated potential reads, after a suitable change of parameters from ε_i to x_i , as

$$V_{model}(\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{2} \sum_i \varepsilon_i^2 \xrightarrow{\varepsilon_i \rightarrow x_i} \frac{1}{2} \sum_i \frac{[\Delta x_i - \Delta t(r_i - \rho_0 x_i)]^2}{\rho_0 \gamma \Delta t x_i^2} + \frac{1}{2} \ln(\rho_0 \gamma x_i^2)$$

where the last term comes out from the Jacobian of the transformation. Notice that this is nothing but the contribution to the posterior given by the discretized path integral associated to the Langevin equation (3.5).

At this point one can choose a suitable time step $\Delta\tau$, propagate the equations of motion for a time window τ and apply the usual HMC algorithm. However, it turns out that this method applied to our specific model leads to very poor performances in terms of efficiency [AUS16]. The reason for this becomes clear when we look at the Hamiltonian and realize that the various terms have dynamics happening at very different time scales. Indeed, the number of discretization points is much larger than the number of observations, $N \gg n \gg 1$, and the scaling of the potentials is

- $V_{model}(\mathbf{x}, \boldsymbol{\theta})$:

$$\begin{aligned} \sum_i \frac{\Delta x_i^2}{\Delta t} &\sim \sum_i \frac{\Delta t}{\Delta t} \sim \mathcal{O}(N) \\ \sum_i \frac{\Delta x_i \Delta t}{\Delta t} &\sim N \Delta x \sim N \sqrt{\Delta t} \sim N \frac{1}{\sqrt{N}} \sim \mathcal{O}(\sqrt{N}) \\ \sum_i \frac{\Delta t^2}{\Delta t} &\sim \sum_i \Delta t \sim N \frac{1}{N} \sim \mathcal{O}(N^0) \end{aligned}$$

- $V_{obs}(\mathbf{x}, \boldsymbol{\theta}; \mathbf{y}_{obs}) \sim \mathcal{O}(n)$

Via a re-parametrization, the $\mathcal{O}(\sqrt{N})$ terms can be reduced to $\mathcal{O}(N^0)$ terms. Thus, upon discretization, the Hamiltonian naturally splits into three terms

$$H = \underbrace{H_N}_{\text{Fast dynamics}} + \underbrace{H_n}_{\text{Intermediate dynamics}} + \underbrace{H_1}_{\text{Slow dynamics}}$$

Vanilla HMC would adjust the discretization time-step $\Delta\tau$ for the Hamiltonian dynamics to the fastest part of the dynamics H_N , which, at the same time, encodes a rather trivial kind of dynamics! In fact, with an

adequate parametrization, this part reduces to N uncoupled harmonic oscillators [AUS16]. Therefore, we would like to allow for different temporal discretizations for the different parts of the dynamics; this is achievable by means of the *Trotter's formula*.

Trotter's formula

From classical mechanics we know that, given an observable function of the positions \mathbf{q} and the momenta \mathbf{p} , $f(\mathbf{q}, \mathbf{p})$, the evolution of the observable under the dynamics generated by the Hamiltonian H is given by

$$\frac{df}{d\tau} = \{f, H\} = \sum_{i=1}^N \left(\frac{\partial f}{\partial q_i} \frac{\partial H}{\partial p_i} - \frac{\partial f}{\partial p_i} \frac{\partial H}{\partial q_i} \right) = -\mathcal{L} f, \quad (3.7)$$

where $\{\cdot, \cdot\}$ indicates the Poisson parentheses whose action can be encapsulated in the operator \mathcal{L} . From the last expression we see that the formal solution of (3.7) is given by

$$f(\tau) = e^{-\tau \mathcal{L}} f(0)$$

If we have an Hamiltonian H which is given by the sum of a fast-dynamics term H_N and a slow-dynamics one H' , then

$$H = H_N + H' \implies \mathcal{L} = \mathcal{L}_N + \mathcal{L}'.$$

Once we introduce the discretization $\tau = P \Delta\tau$, the **Trotter's formula** stands

$$e^{-\tau \mathcal{L}} = \left(e^{-\frac{\Delta\tau}{2} \mathcal{L}_N} e^{-\Delta\tau \mathcal{L}'} e^{-\frac{\Delta\tau}{2} \mathcal{L}_N} \right)^P + \mathcal{O}(\Delta\tau^2) \quad (3.8)$$

which means that we can describe the full dynamics using fast-slow dynamics alternatively (notice that the Trotter's formula is written in terms of operators, not functions). Moreover, we can use a finer discretization to approximate the propagator associated with the fast dynamics.

From now on, to simplify the notation we're going to denote with $\boldsymbol{\pi}$ the momenta and with $\boldsymbol{\theta}$ all the parameters we want to infer.

3.2.2 Riemann Manifold Hamiltonian Monte Carlo

So far we've introduced a kinetic term in the Hamiltonian without caring about the masses. But how can we choose them? Using some physical intuition, we would like to have larger masses where the potential changes rapidly, because the “particles” have to move slower; on the other hand, in the flat and broad regions of the potential a light, fast dynamics is desirable. In other words, we need to have an inertia which depends on the *curvature* of the potential [GC11].

To achieve that, we can generalize our kinetic term with a more general quadratic form

$$K(\boldsymbol{\pi}) = \sum_{i=1}^N \frac{\pi_i^2}{2m_i} \longrightarrow \boldsymbol{\pi}^T g(\boldsymbol{\theta}) \boldsymbol{\pi}$$

where $g(\boldsymbol{\theta})$ is the metric tensor that describes how the curvature of the potential changes depending on where we are. To find out a good candidate for the metric tensor we have first to introduce some kind of “distance” in the parameters' space. One possibility is given by the **relative entropy** (also known as **Kullback-Leibler divergence**)

$$S(\boldsymbol{\theta}, \boldsymbol{\theta}') = KL(f(\mathbf{y}|\boldsymbol{\theta}) || f(\mathbf{y}|\boldsymbol{\theta}')) = \int f(\mathbf{y}|\boldsymbol{\theta}) \ln \frac{f(\mathbf{y}|\boldsymbol{\theta})}{f(\mathbf{y}|\boldsymbol{\theta}')} d\mathbf{y} \quad (3.9)$$

which represents the difference in the information content between $f(\mathbf{y}|\boldsymbol{\theta})$ and $f(\mathbf{y}|\boldsymbol{\theta}')$. Intuitively, since the logarithm of the likelihood has the meaning of a potential, the relative entropy defined above can be also interpreted as the average difference between the potential computed in the two points $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$. If we now expand Eq. (3.9) in Taylor series around $\boldsymbol{\theta}$ we find

$$S(\boldsymbol{\theta}, \boldsymbol{\theta}') = \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}')^T G(\boldsymbol{\theta}) (\boldsymbol{\theta} - \boldsymbol{\theta}') + \mathcal{O}(|\boldsymbol{\theta} - \boldsymbol{\theta}'|^3) \quad (3.10)$$

where

$$G_{ij}(\boldsymbol{\theta}) = - \left\langle \frac{\partial^2 \ln f(\mathbf{y}|\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} \right\rangle_{\boldsymbol{\theta}} = - \int f(\mathbf{y}|\boldsymbol{\theta}) \frac{\partial^2 \ln f(\mathbf{y}|\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} d\mathbf{y} \quad (3.11)$$

is the inverse of the **Fisher metric**.

Exercise Prove Eq. (3.10) and (3.11)

Solution:

We can write Eq. (3.9) as

$$S(\boldsymbol{\theta}, \boldsymbol{\theta}') = \int f(\mathbf{y}|\boldsymbol{\theta}) \ln f(\mathbf{y}|\boldsymbol{\theta}) d\mathbf{y} - \int f(\mathbf{y}|\boldsymbol{\theta}) \ln f(\mathbf{y}|\boldsymbol{\theta}') d\mathbf{y} \quad (3.12)$$

and expand the logarithm of the second term around $\boldsymbol{\theta}$

$$\ln f(\mathbf{y}|\boldsymbol{\theta}') = \ln f(\mathbf{y}|\boldsymbol{\theta}) + (\boldsymbol{\theta}' - \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} \ln f(\mathbf{y}|\boldsymbol{\theta}) + \frac{1}{2} \sum_{ij} (\boldsymbol{\theta}' - \boldsymbol{\theta})_i \frac{\partial^2 \ln f(\mathbf{y}|\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} (\boldsymbol{\theta}' - \boldsymbol{\theta})_j + \mathcal{O}(|\boldsymbol{\theta}' - \boldsymbol{\theta}|^3)$$

By substituting in Eq. (3.12) we get

$$\begin{aligned} S(\boldsymbol{\theta}, \boldsymbol{\theta}') &= - \int f(\mathbf{y}|\boldsymbol{\theta}) (\boldsymbol{\theta}' - \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} \ln f(\mathbf{y}|\boldsymbol{\theta}) d\mathbf{y} + \\ &\quad - \frac{1}{2} \sum_{ij} (\boldsymbol{\theta}' - \boldsymbol{\theta})_i \left\langle \frac{\partial^2 \ln f(\mathbf{y}|\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} \right\rangle_{\boldsymbol{\theta}} (\boldsymbol{\theta}' - \boldsymbol{\theta})_j + \mathcal{O}(|\boldsymbol{\theta}' - \boldsymbol{\theta}|^3) \end{aligned}$$

where we can recognize the Fisher metrics $G_{ij}(\boldsymbol{\theta})$. The first integral splits into a sum whose i -th element is given by

$$\begin{aligned} - \int f(\mathbf{y}|\boldsymbol{\theta}) (\boldsymbol{\theta}' - \boldsymbol{\theta})_i \frac{\partial}{\partial \theta_i} \ln f(\mathbf{y}|\boldsymbol{\theta}) d\mathbf{y} &= - \int f(\mathbf{y}|\boldsymbol{\theta}) (\boldsymbol{\theta}' - \boldsymbol{\theta})_i \frac{1}{f(\mathbf{y}|\boldsymbol{\theta})} \frac{\partial}{\partial \theta_i} f(\mathbf{y}|\boldsymbol{\theta}) d\mathbf{y} \\ &= (\boldsymbol{\theta}' - \boldsymbol{\theta})_i \underbrace{\frac{\partial}{\partial \theta_i} \int f(\mathbf{y}|\boldsymbol{\theta}) d\mathbf{y}}_{=1} = 0 \end{aligned}$$

and we are therefore left with Eq. (3.10)

$$S(\boldsymbol{\theta}, \boldsymbol{\theta}') = \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}')^T G(\boldsymbol{\theta}) (\boldsymbol{\theta} - \boldsymbol{\theta}') + \mathcal{O}(|\boldsymbol{\theta} - \boldsymbol{\theta}'|^3)$$

The idea is that the model $f(\mathbf{y}|\boldsymbol{\theta})$ naturally induces a metrics in the parameter's space which is independent of the observations \mathbf{y}_{obs} and that can be evaluated as the leading term in the relative entropy expansion. The Fisher metrics is also used to fix a lower bound on the error that we make for estimating the parameters of the model.

Theorem 3.1 — Cramer-Rao Given a probability distribution $f(\mathbf{X}|\boldsymbol{\theta})$, whatever unbiased estimator $\hat{\boldsymbol{\theta}}$ of the parameters of the model will have a variance bounded from below by

$$\text{Var}(\hat{\boldsymbol{\theta}}) \geq G(\boldsymbol{\theta})^{-1} = - \left\langle \frac{\partial^2 \ln f(\mathbf{X}|\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} \right\rangle_{\boldsymbol{\theta}}^{-1} \quad (3.13)$$

Exercise Given N i.i.d. random variables drawn from a normal distribution with mean μ and standard deviation σ

$$f(\mathbf{y}|\mu, \sigma) \propto \sigma^{-N} \exp \left[-\frac{1}{2\sigma^2} \sum_{i=1}^N (y_i - \mu)^2 \right],$$

show that

$$G(\mu, \sigma) = \frac{N}{\sigma^2} \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}.$$

This result confirms the well-known fact that the variance of the estimator of the mean of a normal distribution is given by σ^2/N

Solution:

It's easy to see that

$$\begin{aligned}\frac{\partial^2}{\partial \mu^2} \ln f(\mathbf{y}|\mu, \sigma) &= -\frac{N}{\sigma^2}; \\ \frac{\partial^2}{\partial \sigma \partial \mu} \ln f(\mathbf{y}|\mu, \sigma) &= \frac{\partial^2}{\partial \mu \partial \sigma} \ln f(\mathbf{y}|\mu, \sigma) = -\frac{2}{\sigma^3} \sum_{i=1}^N y_i + \frac{2N\mu}{\sigma^3}; \\ \frac{\partial^2}{\partial \sigma^2} \ln f(\mathbf{y}|\mu, \sigma) &= \frac{N}{\sigma^2} - \frac{3}{\sigma^4} \sum_{i=1}^N (y_i - \mu)^2\end{aligned}$$

and taking the average we get

$$\begin{aligned}-\left\langle \frac{\partial^2 \ln f}{\partial \mu^2} \right\rangle &= \frac{N}{\sigma^2} \\ -\left\langle \frac{\partial^2 \ln f}{\partial \mu \partial \sigma} \right\rangle &= -\left\langle \frac{\partial^2 \ln f}{\partial \sigma \partial \mu} \right\rangle = \frac{2}{\sigma^3} \underbrace{\left\langle \sum_{i=1}^N y_i \right\rangle}_{N\mu} - \frac{2N\mu}{\sigma^3} = 0 \\ -\left\langle \frac{\partial^2 \ln f}{\partial \sigma^2} \right\rangle &= -\frac{N}{\sigma^2} + \frac{3}{\sigma^4} \underbrace{\left\langle \sum_{i=1}^N (y_i - \mu)^2 \right\rangle}_{N\sigma^2} = \frac{2N}{\sigma^2}\end{aligned}$$

The Fisher metrics follows. ■

If we have an informative prior, we can replace Eq. (3.11) by

$$\tilde{G}_{ij}(\boldsymbol{\theta}) = -\left\langle \frac{\partial^2 \ln f(\mathbf{y}, \boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} \right\rangle_{\boldsymbol{\theta}} = G_{ij}(\boldsymbol{\theta}) - \frac{\partial^2 \ln f(\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} \quad (3.14)$$

where the additional term is nothing but the Hessian matrix of the prior. We are now tempted to use the inverse of the Fisher matrix as the *mass matrix* of our Hamiltonian, but before doing that we need the following

Proposition 3.2 The Fisher metrics defined as

$$G_{ij}(\boldsymbol{\theta}) = -\left\langle \frac{\partial^2 \ln f(\mathbf{y}|\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} \right\rangle_{\boldsymbol{\theta}}$$

is positive definite.

Proof. Notice that

$$\int f(\mathbf{y}|\boldsymbol{\theta}) \frac{\partial \ln f(\mathbf{y}|\boldsymbol{\theta})}{\partial \theta_i} d\mathbf{y} = \frac{\partial}{\partial \theta_i} \underbrace{\int f(\mathbf{y}|\boldsymbol{\theta}) d\mathbf{y}}_{=1} = 0,$$

thus

$$\begin{aligned}0 &= \frac{\partial}{\partial \theta_j} \int f(\mathbf{y}|\boldsymbol{\theta}) \frac{\partial \ln f(\mathbf{y}|\boldsymbol{\theta})}{\partial \theta_i} d\mathbf{y} = \int \frac{\partial f(\mathbf{y}|\boldsymbol{\theta})}{\partial \theta_j} \frac{\partial \ln f(\mathbf{y}|\boldsymbol{\theta})}{\partial \theta_i} d\mathbf{y} + \underbrace{\int f(\mathbf{y}|\boldsymbol{\theta}) \frac{\partial^2 \ln f(\mathbf{y}|\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} d\mathbf{y}}_{-G_{ij}(\boldsymbol{\theta})} \\ \Rightarrow G_{ij}(\boldsymbol{\theta}) &= \int \frac{\partial f(\mathbf{y}|\boldsymbol{\theta})}{\partial \theta_j} \frac{\partial \ln f(\mathbf{y}|\boldsymbol{\theta})}{\partial \theta_i} d\mathbf{y} = \int f(\mathbf{y}|\boldsymbol{\theta}) \frac{\partial \ln f(\mathbf{y}|\boldsymbol{\theta})}{\partial \theta_j} \frac{\partial \ln f(\mathbf{y}|\boldsymbol{\theta})}{\partial \theta_i} d\mathbf{y} \\ &= \left\langle (\nabla_{\boldsymbol{\theta}} \ln f) (\nabla_{\boldsymbol{\theta}} \ln f)^T \right\rangle_{ij} = \text{Cov}(\nabla_{\boldsymbol{\theta}} \ln f)_{ij}\end{aligned}$$

Since the covariance matrix is positive definite, also $G(\boldsymbol{\theta})$ is positive definite as well. ■

For the case of the likelihood joined with the prior (Eq. (3.14)) we have no guarantee that the Hessian of the prior is positive definite. However this is not typically a problem, since the prior is usually a quite flat distribution way less informative than the likelihood. This means that the first term is predominant, and the total resulting matrix is still positive definite.

By introducing a kinetic term that exploits the inverse of the Fisher metrics as a mass matrix, the posterior can be written as

$$f(\boldsymbol{\theta}|\mathbf{y}) = \frac{\mathcal{Z}^{-1}(\mathbf{y}_{obs})}{\sqrt{\det G(\boldsymbol{\theta})}} \int \exp \left[\ln f(\mathbf{y}|\boldsymbol{\theta}, \mathbf{x}) - \frac{1}{2} \boldsymbol{\pi}^T G^{-1}(\boldsymbol{\theta}) \boldsymbol{\pi} \right] d\mathbf{x} d\boldsymbol{\pi} = \mathcal{Z}^{-1} \int e^{-H(\boldsymbol{\theta}, \mathbf{x}, \boldsymbol{\pi})} d\mathbf{x} d\boldsymbol{\pi}$$

where

$$H(\boldsymbol{\theta}, \mathbf{x}, \boldsymbol{\pi}) = \underbrace{-\ln f(\mathbf{y}|\boldsymbol{\theta}, \mathbf{x})}_{\text{Potential}} + \underbrace{\frac{1}{2} \boldsymbol{\pi}^T G^{-1}(\boldsymbol{\theta}) \boldsymbol{\pi} + \frac{1}{2} \ln \det G(\boldsymbol{\theta})}_{\text{Kinetic term}} \quad (3.15)$$

An example of how this approach works can be taken from the simple likelihood in exercise 3.3. In this case the masses associated to the parameters are

$$m_\mu = \frac{N}{\sigma^2}, \quad m_\sigma = \frac{2N}{\sigma^2}.$$

These expressions are quite reasonable: the more data we have, the more the likelihood is informative (i.e. peaked) and the sharper will be the landscape of the associated potential; this means that heavier masses are desirable. On the other hand, a small value of σ determine a broader likelihood and a quite flat potential, hence requiring lighter “particles”.

Starting from the Hamiltonian (3.15) (and absorbing the input variables \mathbf{x} into the parameters $\boldsymbol{\theta}$), the Hamilton equations become

$$\begin{aligned} \dot{\theta}_i &= \frac{\partial H}{\partial \pi_i} = (G^{-1}(\boldsymbol{\theta}) \boldsymbol{\pi})_i \\ \dot{\pi}_i &= -\frac{\partial H}{\partial \theta_i} = \frac{\partial \ln f(\boldsymbol{\theta}|\mathbf{y})}{\partial \theta_i} + \frac{1}{2} \boldsymbol{\pi}^T \left[G^{-1}(\boldsymbol{\theta}) \frac{\partial G(\boldsymbol{\theta})}{\partial \theta_i} G^{-1}(\boldsymbol{\theta}) \right] \boldsymbol{\pi} - \frac{1}{2} \text{Tr} \left[G^{-1}(\boldsymbol{\theta}) \frac{\partial G(\boldsymbol{\theta})}{\partial \theta_i} \right] \end{aligned}$$

However, with these new Hamilton equations a problem arises: a naive leap-frog discretization will no longer be volume-preserving and time-reversible, and would hence violate detailed balance. For detailed balance to be satisfied, we need an implicit discretization

$$\begin{aligned} \boldsymbol{\pi}(\tau + \Delta\tau/2) &= \boldsymbol{\pi}(\tau) - \frac{\Delta\tau}{2} \nabla_{\boldsymbol{\theta}} H(\boldsymbol{\theta}(\tau), \boldsymbol{\pi}(\tau + \Delta\tau/2)), \\ \boldsymbol{\theta}(\tau + \Delta\tau) &= \boldsymbol{\theta}(\tau) + \frac{\Delta\tau}{2} [\nabla_{\boldsymbol{\pi}} H(\boldsymbol{\theta}(\tau), \boldsymbol{\pi}(\tau + \Delta\tau/2)) + \nabla_{\boldsymbol{\pi}} H(\boldsymbol{\theta}(\tau + \Delta\tau), \boldsymbol{\pi}(\tau + \Delta\tau/2))], \\ \boldsymbol{\pi}(\tau + \Delta\tau) &= \boldsymbol{\pi}(\tau + \Delta\tau/2) - \frac{\Delta\tau}{2} \nabla_{\boldsymbol{\theta}} H(\boldsymbol{\theta}(\tau + \Delta\tau), \boldsymbol{\pi}(\tau + \Delta\tau/2)) \end{aligned}$$

The *duration*, τ , of a HMC update step is also an important parameter to tune. Because the constant-energy level sets are topologically compact, trajectories will eventually return to previously explored neighborhoods. This means that, after a while, samples from that given level set will be no longer informative, and proceeding further with the integration would be just a waste of computational time. Hence, in order to exploit at best these Hamiltonian trajectories, we need to identify the optimal integration time *dynamically*. *No U-turn samplers (NUTS)* detect automatically when a trajectory starts swinging back forth, and stops the simulation. State-of-the-art software for HMC can be downloaded from <http://mc-stan.org>.

Still, if we attempt to sample many coupled degrees of freedom (parameters and system’s realizations) even this kind of algorithm is very slow. In fact, since the parameters are coupled with all the degrees of freedom of the system’s realizations, if we want to make a jump with our parameters we have to drag along the whole system’s realization. This can become fatal if the landscape we want to sample is multi-modal, and the algorithm is so slow that it prevents us to go from one local energy well to the other.

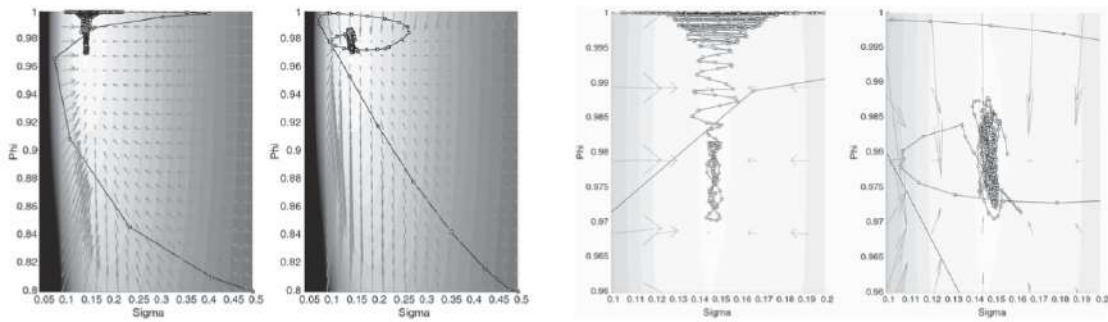


Figure 3.6: Sampling in the parameter's space using vanilla HMC (first and third images from the left) and Riemann manifold HMC (second and fourth images). We can appreciate how the convergence to the target distribution is much faster with Riemann manifold HMC. Figures taken from [GC11]

ABC

4. Approximate Bayesian Computation

4.1 Basic concepts

These methods are applied for model classes for which is rather **cheap** to simulate (sampling with the above methods) an output given a set of parameters, $\mathbf{y} \sim f(\mathbf{y}|\boldsymbol{\theta})$, but it's **expensive** to evaluate the likelihood function for a given set of observations, $f(\mathbf{y}_{obs}|\boldsymbol{\theta})$.

Lesson 6
09/04
LR
AZ

■ **Example 4.1** One of the classes cited above is the one of stochastic differential equation (SDE) models, just as the one we've seen in the previous chapter:

$$x_{i+1} = x_i + \Delta t(r_i - \rho_i x_i), \quad \rho_i = \rho_0 + \sqrt{\frac{\gamma \rho_0}{\Delta t}} \varepsilon_i \quad (4.1)$$

$$y_I = x(t_I) + \sigma \tilde{\varepsilon}_I \quad (4.2)$$

where ε and $\tilde{\varepsilon}$ are random variables following a standard normal distribution. Sampling from this model means:

1. Simulate a time series \mathbf{x} according to the Langevin equation (4.1)
2. Simulate \mathbf{y} given \mathbf{x} using Eq. (4.2)

Thus all in all the sampling procedure is very simple. On the contrary, the likelihood evaluation is totally another business, because we have to compute a marginalization over all the systems' realizations that pass through the data

$$f(\mathbf{y}_{obs}|\boldsymbol{\theta}) = \int f(\mathbf{y}_{obs}|\mathbf{x}, \boldsymbol{\theta}) f(\mathbf{x}|\boldsymbol{\theta}) d\mathbf{x} \quad (4.3)$$

■

■ **Example 4.2** The likelihood function of the Ising model is

$$f(\mathbf{x}|\beta, h) = \mathcal{Z}^{-1}(\beta, h) e^{-\beta H_h(\mathbf{x})} \quad (4.4)$$

where \mathbf{x} is a spin configuration, $\mathcal{Z}(\beta, h)$ is the partition function, β is the inverse temperature and h is an external magnetic field. Also in this case, if we are given β and h is rather simple to generate some realizations, for example by using Gibbs sampling and Metropolis algorithm. Instead, evaluating the likelihood is almost infeasible due to the huge amount of computational resources needed to compute the partition function

$$\mathcal{Z}(\beta, h) = \sum_{\{\mathbf{x}\}} e^{-\beta H_h(\mathbf{x})},$$

where the sum is extended at all possible spin realizations.

■

Approximate Bayesian Computation (ABC) are inference methods based on **model simulation** rather than likelihood evaluation. Basically, what we do is to generate data, sampling them from the likelihood for certain fixed parameters. Then, we compare the simulated dataset with the real data and we decide whether to retain the parameters we used or not. The basic idea follows by the trivial statement that we can write the posterior as

$$f(\boldsymbol{\theta}|\mathbf{y}_{obs}) \propto \int f(\mathbf{y}|\boldsymbol{\theta})f(\boldsymbol{\theta})\delta(\mathbf{y}-\mathbf{y}_{obs})d\mathbf{y} \quad (4.5)$$

where we are conditioning our joint prior of **outputs** and **parameters** (definition of posterior) on the **observations**. From this, we can derive the following preliminary algorithm:

Algorithm 4.1 — Rejection ABC algorithm

1. Sample $\boldsymbol{\theta} \sim f(\boldsymbol{\theta})$
2. Simulate $\mathbf{y} \sim f(\mathbf{y}|\boldsymbol{\theta})$
3. Accept the proposal iff $\mathbf{y} = \mathbf{y}_{obs}$

★

In general, the output of a model is not just apples, bananas and strawberries, and we cannot pretend the simulated data to perfectly match the observations. In other words, we are often dealing with **real numbers** and so we have to relax the last constraint by making two types of approximation:

1. We introduce a **tolerance** ε such that $\mathbf{y} \approx \mathbf{y}_{obs}$.
2. We work with **summary statistics**. Generally, when we have for example a time series, we don't want to compare every element inside, but rather compare only some **features**: $\mathbf{y} \rightarrow \mathbf{s}(\mathbf{y})$ that is a map **from an high to a lower** dimensional space. This is again to avoid the curse of dimensionality.

4.1.1 Summary statistics: basic idea

We're going to go deeper in the important topic of summary statistics later on, but for now it's useful to introduce the concept, in order to better understand what follows. The key point is that we would like to reduce the dimensionality of the variables we're going to deal with, in such a way that the \mathbf{s} retain all the information of \mathbf{y} that are informative about the parameters and that throws away all the rest, which is just noise. Mathematically, the requirement for $\mathbf{s}(\mathbf{y})$ to contain all the information about parameters $\boldsymbol{\theta}$ translates into the concept of **sufficiency**

$$f(\mathbf{y}|\boldsymbol{\theta}) = c(\mathbf{y})g(\mathbf{s}(\mathbf{y}),\boldsymbol{\theta}),$$

where we multiply a function that depends on the data only through summary statistics with a term that does not depend on parameters. This is because, for the sake of inference, we are only interested in how f depends on $\boldsymbol{\theta}$. If the part of the function that depends on $\boldsymbol{\theta}$ only depends on \mathbf{s} , then it's clear that this piece alone contains all we want to know about our parameters. Obviously, $\mathbf{s}(\mathbf{y}) = \mathbf{y}$, i.e. the identity, is always a sufficient statistics, but we want a map in a lower dimensional space.

■ **Example 4.3** Suppose $y_1, \dots, y_N \sim \mathcal{N}(\mu, \sigma)$ i.i.d., so that effortlessly

$$\begin{aligned} f(\mathbf{y}|\mu, \sigma) &\propto \sigma^{-N} \exp \left[-\frac{1}{2\sigma^2} \sum_i (y_i - \mu)^2 \right] \\ &= \sigma^{-N} \exp \left[-\frac{1}{2\sigma^2} \underbrace{\sum_i y_i^2}_{S_2(\mathbf{y})} + \frac{\mu}{\sigma^2} \underbrace{\sum_i y_i}_{S_1(\mathbf{y})} - \frac{N\mu^2}{2\sigma^2} \right] = g(\mathbf{s}(\mathbf{y}), \boldsymbol{\theta}) \end{aligned}$$

So if we have a large set of observations and we want to infer $\boldsymbol{\theta} = (\mu, \sigma)$, it is sufficient to consider only two summary statistics, for example the mean and the empirical variance. Another example is given by the Ising model, where we can try to infer the temperature for a given configuration just by computing the energy and the magnetization. However, in general, finding such low dimensional $\mathbf{s}(\mathbf{y})$ is hard, and indeed it's a typical **machine learning problem**. ■

4.2 Tolerance: the SABC algorithm

Now we shall focus on finding a good **tolerance** ε . The choice of the tolerance, i.e. what we mean by having a good fit between observations and simulations, is a crucial point. In fact, if we choose ε too large we will

end up with a large bias in our posterior, while a parameter too small will never make the algorithm converge. Inspired by thermodynamics, we can think to interpret this tolerance as a temperature, so that the metric that we use to measure the distance between simulated and observed outputs, $\rho(\mathbf{y}) := \rho(\mathbf{y}, \mathbf{y}_{obs})$, will correspond to an energy. Indeed, what we strive for is to approximate our posterior in the form of a sort of Gibbs' state

$$\pi_T(\mathbf{y}, \boldsymbol{\theta}) = f(\mathbf{y}, \boldsymbol{\theta}) \exp \left\{ -\frac{\rho(\mathbf{y}, \mathbf{y}_{obs})}{T} \right\} \quad (4.6)$$

where, as usual, $f(\mathbf{y}, \boldsymbol{\theta}) = f(\mathbf{y}|\boldsymbol{\theta})f(\boldsymbol{\theta})$. Notice that here we write in the measure the explicit dependence on \mathbf{y} and \mathbf{y}_{obs} , but we can actually think of it to depend only on the summary statistics.

The idea now is to use **simulated annealing** [AKS14], a procedure widely used in optimization algorithms, where annealing simply means reducing gradually the temperature. In terms of inference, this means that we gradually move from the prior to the posterior. In fact, if in Eq. (4.6) we take $T \rightarrow \infty$ then we have simply the joint prior of outputs and parameters, while if we take $T \rightarrow 0$ the Gibbs factor reduces to a Dirac's delta centered in \mathbf{y}_{obs} , and we're left with the definition of the posterior.

Consider now figure 4.1, and think of the green area as a system composed of particles in the product space of parameters and model's outputs. The environment is represented by our computer, and it is parameterized by the temperature T^e . Each computer update consists on asking a question to our system, and the result can be seen as a flow of entropy (and heat) from the system to the environment. Part of this entropy flux is well-invested, because it allows us to get from the wide prior to the narrow posterior, but part of it is just **wasted computation**. This waste is quantified as a unavoidable **entropy production** due to the **irreversibility** of the process. Our goal is to define this problem, circumscribe and quantify the waste and try to minimize it, in order to get as closer to the posterior as possible with the minimum computational effort.

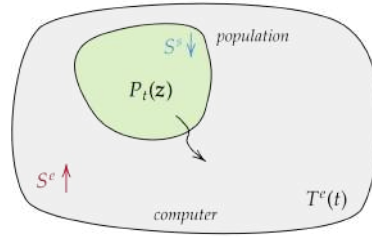


Figure 4.1: We move our prior population closer to the posterior, by continuously addressing questions to the system to gain information. The entropy of the system decreases in time while the one of the computer does the opposite.

This idea is implemented by seeing the system as a gas of particles, where each particle has two components $\{z_i := (\boldsymbol{\theta}_i, \mathbf{y}_i)\} \sim P_t(\mathbf{z})$ and is drawn from a distribution that we will call $P_t(\mathbf{z})$. We will have to make sure that these particles represent $P(\mathbf{z}) \approx \pi_{T_f}(\mathbf{z})$, i.e., our final distribution. Let's now describe the process.

Lesson 7
12/04
TF
AM

Algorithm 4.2 — SABC The algorithm breaks down in the following steps (the details of the algorithm can be found in [AKS14]):

1. Pick a random particle from the population, \mathbf{z}_i ;
2. **Jump**, first in positions, drawing $\boldsymbol{\theta}^* \sim t(\boldsymbol{\theta}|\boldsymbol{\theta}_i)$;
3. Simulate $\mathbf{y}^* \sim f(\mathbf{y}|\boldsymbol{\theta}^*)$;
4. Accept $(\boldsymbol{\theta}^*, \mathbf{y}^*)$ with a probability of $\min \left(1, \frac{f(\boldsymbol{\theta}^*)}{f(\boldsymbol{\theta}_i)} \exp \left\{ \frac{-\rho(\mathbf{y}^*) + \rho(\mathbf{y}_i)}{T^e(t)} \right\} \right)$ where the exponential is the Boltzmann factor.
5. We **lower the temperature** $T^e(t)$ a bit.

★

Exercise Check detailed balance.

Let's firstly observe that if the temperature is kept constant in time, i.e. $T^e(t) = T^e$, then the detailed balance condition is satisfied and so

$$P_t(\mathbf{z}) \xrightarrow[t \rightarrow \infty]{} \pi_{T^e}(\mathbf{z}) = f(\mathbf{z})e^{-\frac{\rho(\mathbf{z})}{T^e}}$$

i.e. the particles distribution converges to our target, where $f(\mathbf{z}) = f(\mathbf{y}|\boldsymbol{\theta})f(\boldsymbol{\theta})$ is the joint prior (notice that we can write either \mathbf{z} or \mathbf{y} : always remember that the distance depends on the \mathbf{y} component of the particle). Notice also that if we keep the temperature sufficiently low we have that the Boltzmann factor will be very sharply concentrated around the observations and will condition our joint prior $f(\mathbf{z})$ to \mathbf{y}_{obs} .

Let's consider now an environment temperature that is evolving in time $T^e(t)$: in this case the dynamic of the system under examination (given that it corresponds to a **sufficiently large population**) is described by the **Master Equation** (ME)

$$\frac{\partial P_t(\mathbf{z})}{\partial t} = \int P_t(\mathbf{z}|\mathbf{z}')P_t(\mathbf{z}') - P_t(\mathbf{z}'|\mathbf{z})P_t(\mathbf{z})d\mathbf{z}' \quad (4.7)$$

in which we define $P_t(\mathbf{z}'|\mathbf{z})$ as the **transition probability**

$$P_t(\mathbf{z}'|\mathbf{z}) = p_t(\mathbf{z}', \mathbf{z}) + \delta(\mathbf{z}' - \mathbf{z}) \left(1 - \int p_t(\mathbf{z}', \mathbf{z})d\mathbf{z}' \right) \quad (4.8)$$

with

$$p_t(\mathbf{z}', \mathbf{z}) = t(\boldsymbol{\theta}'|\boldsymbol{\theta})f(\mathbf{y}'|\boldsymbol{\theta}') \min \left(1, \frac{f(\boldsymbol{\theta}')}{f(\boldsymbol{\theta})} e^{-\frac{\rho(\mathbf{y}') + \rho(\mathbf{y})}{T^e(t)}} \right). \quad (4.9)$$

Here, $t(\boldsymbol{\theta}'|\boldsymbol{\theta})$ is the **symmetric** jump probability, while the second term in Eq. (4.8) can be interpreted as the probability to reject the move.

The meaning of (4.7) is clear: we have the probability to jump in state \mathbf{z} from \mathbf{z}' minus the one of jumping away from it. But now the question is: *how do we lower T^e* ? We need to be aware of the fact that, if in the extreme case T^e is 0 from the beginning, then any update state away from the target is surely rejected (look at the formula in point 4 of the algorithm), and if instead $T^e = \infty$ then \mathbf{y}_{obs} does not play any role. Summarizing, if we lower the temperature too fast we get a bias in the final posterior estimate, if we lower the temperature too slow we are just wasting computational time. The way in which the temperature is lowered is one of the most crucial points of the algorithm. In order to accomplish this task there exist several ways: one can follow a **deterministic** schedule or an **adaptive** schedule. The first one consists in lowering the temperature slower than a certain power law $T^e(t) > ct^{-\frac{\alpha}{n}}$, and there is a theorem that ensures the desired convergence:

Theorem 4.1 — (AKS14) Choosing as a distance

$$\rho(\mathbf{y}, \mathbf{y}_{obs}) = \frac{1}{\alpha} \sum_{i=1}^n |y_i - y_{obs,i}|^\alpha, \quad \alpha > 0$$

then, for

$$T^e(t) > ct^{-\alpha/n}$$

we have that

$$\int P_t(\mathbf{z})d\mathbf{y} \xrightarrow[t \rightarrow \infty]{} f(\boldsymbol{\theta}|\mathbf{y}_{obs})$$

where the convergence is in the total variation (TV) sense.

As we can deduce, the larger is n (the dimension of the output space), the slower we are allowed to reduce temperature in order to be sure to converge to the right result (indeed in the limit we have $T^e(t) \xrightarrow{n \rightarrow \infty} > c$). Notice that this theorem doesn't tell **anything about the simulation time** of the algorithm (if we early stop we don't know where we are). This is a deterministic schedule, but rather we are interested in an adaptive schedule where we **adapt T^e to the average distance** of \mathbf{y}_i to \mathbf{y}_{obs} , and we want to do it in such a way that S_{prod} is minimized. It's weird to say, but good algorithms are not always supported with a rigorous convergence proof. This is indeed the case of SABC with an adaptive temperature decrease.

In a nutshell: Roughly speaking: we must find a way to update the environmental temperature and we can do it in two ways. For the deterministic approach it exists a theorem stating that convergence can be achieved if a certain distance in variable space is exploited, but no information about which α parameter and how many iterations are needed is reported. As a consequence, the deterministic way may result really computational demanding because of the necessary trial to calibrate the procedure. On the other hand, the adaptive approach is not formally proven to come to convergence but direct experience confirms good results. Moreover, we can convince ourselves of the quality of this method according to an heuristic point of view (see next paragraphs).

4.2.1 Adaptive schedule

Let's now switch to an adaptive schedule and discuss an heuristic proof of converge of the algorithm. Recalling Fig. 4.1, we can write the **total invested entropy** as

$$S_{env} = \underbrace{[S(P_{t_i}) - S(P_{t_f})]}_{>0} + S_{prod} \quad (4.10)$$

The term between squared brackets (difference between entropy for the prior and the posterior) is usually greater than 0 since the prior is broader (and thus "less informative") than the posterior. This is also the term quantifying the right amount of entropy that should flow from the system to the environment in order to have optimal information transmission: the last term represents the ulterior entropy that the environment gains, which is not exploited and results in a waste of calculations. As we already said above, the aim is to minimize this **entropy of production**. Mathematically speaking, for a given annealing schedule in a number n of discrete time-steps T_n^e , S_{prod} is defined on the **space of paths** $\Gamma_n = (\mathbf{z}_0, \mathbf{z}_1, \dots, \mathbf{z}_{n-1})$ [Sei05]

$$S_{prod}(n) = \int P(\Gamma_n) \ln \frac{P(\Gamma_n)}{P^R(\Gamma_n)} d\Gamma_n \quad (4.11)$$

where

- $P(\Gamma_n) = P_{n-2}(\mathbf{z}_{n-1}|\mathbf{z}_{n-2}) \dots P_0(\mathbf{z}_1|\mathbf{z}_0)P_0(\mathbf{z}_0)$ is the forward probability of seeing a path realized (direct, annealing)
- $P^R(\Gamma_n) = P_0(\mathbf{z}_0|\mathbf{z}_1) \dots P_{n-2}(\mathbf{z}_{n-2}|\mathbf{z}_{n-1})P_{n-1}(\mathbf{z}_{n-1})$ is the backward probability of seeing the same path realized (reverse, heating).

In the expression above,

$$P_{n-1}(\mathbf{z}_{n-1}) = \int P(\Gamma_n) d\mathbf{z}_0 \dots d\mathbf{z}_{n-2}$$

is the distribution of the final state.

Entropy of production is therefore a **measure of irreversibility**, that under detailed balance is 0. We can rearrange Eq. (4.10) as

$$S_{prod}(n) = S(P_{n-1}) - S(P_0) + S_{env},$$

hence, if $P_0(\mathbf{z}) = f(\mathbf{y}, \boldsymbol{\theta})$, the entropy production is the difference between the entropy reduction in the system (difference between the approximate posterior entropy and the prior entropy) and the entropy increase in the environment (number of questions asked through computer updates). The entropy difference between two steps reads

$$\Delta S_{prod} = S_{prod}(n+1) - S_{prod}(n) = \int d\mathbf{z}_{n-1} d\mathbf{z}_n \ln \frac{P_{n-1}(\mathbf{z}_n|\mathbf{z}_{n-1})P_{n-1}(\mathbf{z}_{n-1})}{P_{n-1}(\mathbf{z}_{n-1}|\mathbf{z}_n)P_n(\mathbf{z}_n)} P_{n-1}(\mathbf{z}_n|\mathbf{z}_{n-1})P_{n-1}(\mathbf{z}_{n-1}) \quad (4.12)$$

It's time now to go back to our particle population point of view. In the large population limit (move to a continuous representation), we pass from probability to transition rates

$$\dot{S}_{prod}(t) = \int d\mathbf{z} d\mathbf{z}' \ln \frac{p_t(\mathbf{z}', \mathbf{z})P_t(\mathbf{z})}{p_t(\mathbf{z}, \mathbf{z}')P_t(\mathbf{z}')} p_t(\mathbf{z}', \mathbf{z})P_t(\mathbf{z}) \quad (4.13)$$

Exercise Prove the last equation

Solution:

In the continuum limit we can denote $\mathbf{z} \equiv \mathbf{z}_{n-1}$ and $\mathbf{z}' \equiv \mathbf{z}_n$. Eq. (4.12) can be split as

$$\dot{S}_{prod}(t) = \underbrace{\int \ln \frac{P_{n-1}(\mathbf{z}'|\mathbf{z})}{P_{n-1}(\mathbf{z}|\mathbf{z}')} P_{n-1}(\mathbf{z}'|\mathbf{z}) P_{n-1}(\mathbf{z}) d\mathbf{z} d\mathbf{z}'}_{(a)} + \underbrace{\int \ln \frac{P_{n-1}(\mathbf{z})}{P_n(\mathbf{z}')} P_{n-1}(\mathbf{z}'|\mathbf{z}) P_{n-1}(\mathbf{z}) d\mathbf{z} d\mathbf{z}'}_{(b)}$$

In the first term, when $\mathbf{z} = \mathbf{z}'$ the ratio in the logarithm is 1, and therefore this contribution is zero. From Eq. (4.8) we therefore see that we can write $p_t(\mathbf{z}', \mathbf{z})$ in place of $P_{n-1}(\mathbf{z}'|\mathbf{z})$, where we also identify $t_{n-1} \equiv t$. Hence

$$(a) = \int \frac{p_t(\mathbf{z}', \mathbf{z})}{p_t(\mathbf{z}, \mathbf{z}')} p_t(\mathbf{z}', \mathbf{z}) P_t(\mathbf{z}) d\mathbf{z} d\mathbf{z}'$$

The (b) term can be further split as

$$(b) = \underbrace{\int P_{n-1}(\mathbf{z}'|\mathbf{z}) P_{n-1}(\mathbf{z}) \ln P_{n-1}(\mathbf{z}) d\mathbf{z} d\mathbf{z}'}_{(c)} - \underbrace{\int P_{n-1}(\mathbf{z}'|\mathbf{z}) P_{n-1}(\mathbf{z}) \ln P_n(\mathbf{z}') d\mathbf{z} d\mathbf{z}'}_{(d)}$$

where

$$(c) = \int \left(\underbrace{\int P_{n-1}(\mathbf{z}'|\mathbf{z}) d\mathbf{z}'}_{=1} \right) P_{n-1}(\mathbf{z}) \ln P_{n-1}(\mathbf{z}) d\mathbf{z} = \int P_{n-1}(\mathbf{z}) \ln P_{n-1}(\mathbf{z}) d\mathbf{z}$$

$$(d) = \int \left(\underbrace{\int P_{n-1}(\mathbf{z}'|\mathbf{z}) P_{n-1}(\mathbf{z}) d\mathbf{z}}_{=P_n(\mathbf{z}')} \right) \ln P_n(\mathbf{z}') d\mathbf{z}' \stackrel{\mathbf{z} \leftrightarrow \mathbf{z}'}{=} \int P_n(\mathbf{z}) \ln P_n(\mathbf{z}) d\mathbf{z}$$

Thus

$$(b) = (c) - (d) = -\frac{d}{dt} \int \ln P_t(\mathbf{z}) P_t(\mathbf{z}) d\mathbf{z} = -\frac{\partial P_t(\mathbf{z})}{\partial t} \frac{\partial}{\partial P_t(\mathbf{z})} \int \ln P_t(\mathbf{z}) P_t(\mathbf{z}) d\mathbf{z}$$

$$= -\frac{\partial P_t(\mathbf{z})}{\partial t} \int 1 + \ln P_t(\mathbf{z}) d\mathbf{z} = -\int \ln P_t(\mathbf{z}) \frac{\partial P_t(\mathbf{z})}{\partial t} d\mathbf{z}$$

Now we can substitute the master equation (4.7), obtaining

$$(b) = -\int \ln P_t(\mathbf{z}) \left[\int p_t(\mathbf{z}, \mathbf{z}') P_t(\mathbf{z}') - p_t(\mathbf{z}', \mathbf{z}) P_t(\mathbf{z}) d\mathbf{z}' \right] d\mathbf{z}$$

$$= -\int \ln P_t(\mathbf{z}) p_t(\mathbf{z}, \mathbf{z}') P_t(\mathbf{z}') d\mathbf{z} d\mathbf{z}' + \int \ln P_t(\mathbf{z}) p_t(\mathbf{z}', \mathbf{z}) P_t(\mathbf{z}) d\mathbf{z} d\mathbf{z}'$$

In the first term we can exchange $\mathbf{z} \leftrightarrow \mathbf{z}'$, and joining all the pieces together we end up with

$$\dot{S}_{prod}(t) = \int (\ln P_t(\mathbf{z}) - \ln P_t(\mathbf{z}')) p_t(\mathbf{z}', \mathbf{z}) P_t(\mathbf{z}) d\mathbf{z} d\mathbf{z}'$$

$$= \int \ln \frac{p_t(\mathbf{z}', \mathbf{z}) P_t(\mathbf{z})}{p_t(\mathbf{z}, \mathbf{z}') P_t(\mathbf{z}')} p_t(\mathbf{z}', \mathbf{z}) P_t(\mathbf{z}) d\mathbf{z} d\mathbf{z}'$$

We have a little information on how the distribution behaves, so we make two assumptions:

- **Endo-reversibility assumption.** At any given point in time, a system under the endo-reversibility assumption is always in an equilibrium state, i.e. it is described by a Gibbs state

$$P_t(\mathbf{z}) = Z^{-1}(T(t)) f(\mathbf{y}, \boldsymbol{\theta}) e^{-\frac{\rho(\mathbf{y})}{T(t)}} \quad (4.14)$$

where $T(t) \gtrsim T^e(t)$ is the system's temperature, that is slightly larger than the temperature of the environment. The latter is defined by the **tolerance** used for the update step and subject to a deter-

ministic or adaptive annealing schedule. Intuitively, this assumption is satisfied if there is sufficiently fast mixing in the system. In our setting, mixing is determined by the width of the jump distribution $t(\boldsymbol{\theta}'|\boldsymbol{\theta})$, which should be as large as the typical scale of particles movements. Please notice that if this assumption is not satisfied, then we incur in a bias (no convergence to the solution).

- **A weaker assumption.** We assume that the prior $f(\boldsymbol{\theta})$ carries negligible information, i.e., we set it equal to a uniform (constant) distribution. This is often the case since, as compared to the prior, the posterior is narrower. Furthermore, the annealing techniques tends to work better when the prior is **flat**.

By inserting Eq. (4.9) into Eq. (4.13) and using the endo-reversibility assumption, we get the following expression for the entropy production rate

$$\dot{S}_{prod} = \underbrace{\left(\frac{1}{T} - \frac{1}{T^e(t)} \right)}_{F(t)} \underbrace{\frac{d}{dt} \int \rho(\mathbf{z}) P_t(\mathbf{z}) d\mathbf{z}}_{U(t)} = F(t) \dot{U}(t)$$

where $F(t)$ is the *thermodynamic force* and $\dot{U}(t)$ the *thermodynamic heat flux* from the system.

Exercise Prove the previous equation

Solution:

Let's start focusing on the logarithm of

$$\dot{S}_{prod}(t) = \int d\mathbf{z} d\mathbf{z}' \ln \frac{p_t(\mathbf{z}', \mathbf{z}) P_t(\mathbf{z})}{p_t(\mathbf{z}, \mathbf{z}') P_t(\mathbf{z}')} p_t(\mathbf{z}', \mathbf{z}) P_t(\mathbf{z}) \quad (4.15)$$

Substituting the expression for the transition density (4.9), we have

$$\ln \left[\frac{t(\boldsymbol{\theta}'|\boldsymbol{\theta}) f(\mathbf{y}'|\boldsymbol{\theta}') \min \left(1, \frac{f(\boldsymbol{\theta}')}{f(\boldsymbol{\theta})} \exp \left(\frac{\rho(\mathbf{y}) - \rho(\mathbf{y}')}{T^e(t)} \right) \right) P_t(\mathbf{z})}{t(\boldsymbol{\theta}|\boldsymbol{\theta}') f(\mathbf{y}|\boldsymbol{\theta}) \min \left(1, \frac{f(\boldsymbol{\theta})}{f(\boldsymbol{\theta}')} \exp \left(\frac{\rho(\mathbf{y}') - \rho(\mathbf{y})}{T^e(t)} \right) \right) P_t(\mathbf{z}')} \right] =$$

where we could cancel out the transition probability under the assumption that it is symmetric. For the moment, let's assume that the Metropolis ratio at numerator is smaller than 1. Since the corresponding term at denominator is just its reciprocal, this means that the latter will be greater than 1. That being said, we can write

$$\ln \left[\frac{f(\mathbf{y}'|\boldsymbol{\theta}') f(\boldsymbol{\theta}')}{f(\mathbf{y}|\boldsymbol{\theta}) f(\boldsymbol{\theta})} \exp \left(\frac{\rho(\mathbf{y}) - \rho(\mathbf{y}')}{T^e(t)} \right) \frac{f(\mathbf{y}, \boldsymbol{\theta})}{f(\mathbf{y}', \boldsymbol{\theta}')} \exp \left(-\frac{\rho(\mathbf{y}) - \rho(\mathbf{y}')}{T(t)} \right) \right] =$$

$$\left(\frac{1}{T(t)} - \frac{1}{T^e(t)} \right) (\rho(\mathbf{y}') - \rho(\mathbf{y}))$$

It's straightforward to see that also the case in which the Metropolis ratio at numerator is greater than 1 and the other is smaller than one leads to the same result. Eq. (4.15) becomes

$$\dot{S}_{prod}(t) = \left(\frac{1}{T(t)} - \frac{1}{T^e(t)} \right) \int d\mathbf{z} d\mathbf{z}' (\rho(\mathbf{z}') - \rho(\mathbf{z})) p_t(\mathbf{z}', \mathbf{z}) P_t(\mathbf{z})$$

$$= \left(\frac{1}{T(t)} - \frac{1}{T^e(t)} \right) \left[\int d\mathbf{z}' \rho(\mathbf{z}') \int d\mathbf{z} p_t(\mathbf{z}', \mathbf{z}) P_t(\mathbf{z}) - \int d\mathbf{z} \rho(\mathbf{z}) \int d\mathbf{z}' p_t(\mathbf{z}', \mathbf{z}) P_t(\mathbf{z}) \right]$$

If we change $\mathbf{z}' \leftrightarrow \mathbf{z}$ in the first term, we get

$$\dot{S}_{prod}(t) = \left(\frac{1}{T(t)} - \frac{1}{T^e(t)} \right) \int d\mathbf{z} \rho(\mathbf{z}) \left[\int p_t(\mathbf{z}, \mathbf{z}') P_t(\mathbf{z}') - p_t(\mathbf{z}', \mathbf{z}) P_t(\mathbf{z}) d\mathbf{z}' \right]$$

$$\stackrel{(*)}{=} \left(\frac{1}{T(t)} - \frac{1}{T^e(t)} \right) \int d\mathbf{z} \rho(\mathbf{z}) \frac{dP_t(\mathbf{z})}{dt} = \left(\frac{1}{T(t)} - \frac{1}{T^e(t)} \right) \frac{d}{dt} \int \rho(\mathbf{z}) P_t(\mathbf{z}) d\mathbf{z}$$

where in (*) we used the master equation (4.7) ■

Since the entropy production represents the wasted computation of the algorithm, in order to move

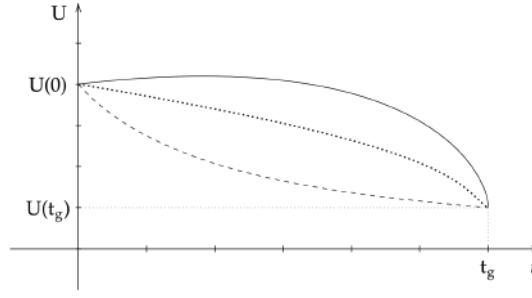


Figure 4.2: Possible annealing schedules

towards convergence it is necessary to minimize it, i.e., make its derivative vanish. One must take into account that the system could explore multiple paths from an initial to a final energy state in time, thus we exploit variational calculus to collect significant results. If we impose boundaries conditions at the initial and final state (namely $\delta U_0 = \delta U_f = 0$) and we notice that in this thermodynamical approach we can find a one-to-one correspondence between $T \iff U$ and $T, T^e \iff \dot{U}$ (see next paragraphs), we have:

$$\begin{aligned} \delta S_{prod} &= \delta \left(\int_0^{t_f} \dot{S}_{prod} dt \right) = \delta \left(\int_0^{t_f} F(T, T^e) \dot{U} dt \right) = \delta \left(\int_0^{t_f} F(U, \dot{U}) \dot{U} dt \right) = \\ &= \int_0^{t_f} \left(\frac{\partial F}{\partial U} \delta U \dot{U} + \frac{\partial F}{\partial \dot{U}} \delta \dot{U} \dot{U} + F \delta \dot{U} \right) dt \stackrel{(*)}{=} \int_0^{t_f} \delta U \left(\frac{\partial F}{\partial U} \dot{U} - \frac{d}{dt} \left(\frac{\partial F}{\partial \dot{U}} \dot{U} + F \right) \right) dt \end{aligned}$$

where in (*) we've applied an integration by parts to the last two terms. From the previous equations we can conclude that if we want to minimize the entropy production (and therefore its variation vanishes), then:

$$\delta S_{prod} = 0 \forall \delta U \iff \frac{\partial F}{\partial U} \dot{U} - \frac{d}{dt} \left(\frac{\partial F}{\partial \dot{U}} \dot{U} + F \right) = 0$$

This last requirement is in the form of the Euler-Lagrange equations and expanding the last term one obtains

$$\begin{aligned} 0 &= \cancel{\frac{\partial F}{\partial U} \dot{U}} - \frac{d}{dt} \left(\frac{\partial F}{\partial \dot{U}} \right) \dot{U} - \frac{\partial F}{\partial \dot{U}} \ddot{U} - \cancel{\frac{\partial F}{\partial U} \dot{U}} - \frac{\partial F}{\partial \dot{U}} \ddot{U} \\ \stackrel{(*)}{\iff} 0 &= \left(\frac{d}{dt} \left(\frac{\partial F}{\partial \dot{U}} \right) \dot{U} + 2 \frac{\partial F}{\partial \dot{U}} \ddot{U} \right) \dot{U} = \frac{d}{dt} \left(\frac{\partial F}{\partial \dot{U}} \right) \dot{U}^2 + 2 \frac{\partial F}{\partial \dot{U}} \dot{U} \ddot{U} = \frac{d}{dt} \left(\dot{U}^2 \frac{\partial F}{\partial \dot{U}} \right) \end{aligned}$$

Where in (*) we multiplied by a \dot{U} factor and absorbed a $(-)$ sign.

As a consequence of the previous result, we can thus conclude that the minimization on the entropy of production corresponds with the time conservation of the product of the squared time derivative of the internal energy, multiplied by the derivative of the force w.r.t. the time derivative of the energy itself:

$$\delta S_{prod} = 0 \forall \delta U \iff \dot{U}^2 \frac{\partial F}{\partial \dot{U}} =: v = const. \quad (4.16)$$

The v parameter in equation (4.16) is defined the **adaptive annealing speed** and is one of the parameters to be tuned for this algorithm.

Formalization of $T \iff U$ and $T, T^e \iff \dot{U}$

In the previous calculations we exploited the fact that there exist two direct correspondences between energy and temperatures: we now aim to formalize them. Firstly, we recall that is has not been defined so far the distance $\rho(\mathbf{y})$ between the generated variables and the observed data. Hence, we attempt to choose it such that the system has a constant heat capacity, i.e., such that

$$U(t) \approx T(t)$$

as requested previously. This can be achieved by redefining the energy function as the cumulative density function of the newly user-specified distance measure $\rho(\mathbf{y})$ under the prior $f(\mathbf{y}, \boldsymbol{\theta})$:

$$u(\mathbf{y}) := \int_{\rho(\mathbf{y}') \leq \rho(\mathbf{y})} f(\mathbf{y}', \boldsymbol{\theta}) d\boldsymbol{\theta} d\mathbf{y}' \quad (4.17)$$

At the beginning, in the initialization phase of the algorithm, we draw a large number of particles from the prior, so the previous equation can be approximated to

$$u(\mathbf{y}) \approx \# \{ \mathbf{z}_i = (\mathbf{y}_i, \boldsymbol{\theta}_i) \sim f(\mathbf{z}) \mid \rho(\mathbf{y}_i) \leq \rho(\mathbf{y}) \},$$

and from here it is easy to calculate the mean energy of the system under f :

$$U(t) = \langle u \rangle = Z^{-1}(t) \int u(\mathbf{z}) f(\mathbf{z}) \exp\left(-\frac{u(\mathbf{z})}{T}\right) d\mathbf{z}$$

Finally, one can notice that $d\mathbf{z} \equiv d\boldsymbol{\theta} d\mathbf{y}$ by definition of \mathbf{z} and $d\mathbf{y} \equiv d\rho d\Omega(\rho)$ by moving to spherical coordinates and identifying the radial and superficial contributions, thus the following holds

$$\int f(\mathbf{z}) d\boldsymbol{\theta} d\Omega(\rho) = \frac{du}{d\rho}$$

Notice that this derivative gives the prior measure in a shell of radius ρ . So, by substitution in the previous equation one can conclude:

$$U(t) = Z^{-1}(t) \int_0^1 u \exp\left(-\frac{u}{T(t)}\right) du = \frac{\int_0^1 u \exp\left(-\frac{u}{T(t)}\right) du}{\int_0^1 \exp\left(-\frac{u}{T(t)}\right) du} = T(t) + o(T(t)^k) \quad \forall k > 0 \quad (4.18)$$

where the normalization is given by the partition function.

To conclude, one can be easily convinced that the last expression approximated at the first order in Taylor's expansions returns:

$$U(t) = T(t) + h.o. \quad (4.19)$$

where with higher orders we means exponentially small terms.

The previous result confirms the one-to-one relation existing between energy and temperature. We now aim to find a similar expression for the correspondence $\dot{U} \leftrightarrow T, T^e$. In fact, we can write:

$$\dot{U} \equiv \frac{dU}{dt} = \frac{d}{dt} \int u(\mathbf{z}) P_t(\mathbf{z}) d\mathbf{z} \stackrel{(4.7)}{=} \int (u(\mathbf{z}) - u(\mathbf{z}')) p_t(\mathbf{z}, \mathbf{z}') P_t(\mathbf{z}') d\mathbf{z} d\mathbf{z}'$$

One can solve the latter expression by using Taylor's series approximation at the second order around u and u' , given the fact that re-normalized energies are $\ll 1$ for most of the process and after some tedious calculations that go beyond the goal of this course we can conclude:

$$\dot{U} \stackrel{1^{st} \text{ order}}{=} -\gamma \left(T^2 - (T^e)^2 \right) \implies T^e = \sqrt{U^2 + \frac{\dot{U}}{\gamma}} \quad (4.20)$$

Where the γ factor is a constant value setting the proportionality.

4.2.2 SABC tunable parameters

We finally summarize the tunable parameters of this algorithm. We have already shown that v is one of these, but its form makes it hard to calculate directly. Nevertheless, recalling equation (4.16) we have:

$$v = \dot{U}^2 \frac{\partial F}{\partial \dot{U}} = \gamma^2 \left[U^2 - (T^e)^2 \right]^2 \left[\frac{\partial}{\partial \dot{U}} \left(\frac{1}{T} - \frac{1}{T^e} \right) \right] = \frac{\gamma \left[U^2 - (T^e)^2 \right]^2}{2(T^e)^3}$$

$$\text{where } \gamma = (f(\mathbf{y}_{obs}))^{-2} \int t(\boldsymbol{\theta} | \boldsymbol{\theta}') f(\mathbf{y}_{obs}, \boldsymbol{\theta}) f(\mathbf{y}_{obs}, \boldsymbol{\theta}') d\boldsymbol{\theta} d\boldsymbol{\theta}'$$

Thus, since both v and γ are unknown parameters of the algorithm, we redefine a unique *effective annealing speed* \tilde{v} :

$$\tilde{v} := \frac{v}{\gamma} = \frac{\left[U^2 - (T^e)^2 \right]^2}{2(T^e)^3} \quad (4.21)$$

Since \tilde{v} is an hyperparameter set by the user and the internal energy U is an observable quantity of the population (average distance from the target w.r.t the normalized metric), solving Eq. (4.21) for T^e then gives us the annealing temperature for the next update step, i.e.:

$$(T^e)^4 + 2\tilde{v}(T^e)^3 + 2U^2(T^e)^2 - U^4 = 0$$

Finally, one can see that for small U we encounter a dimension-independent scaling $T^e \sim U^{4/3}$, which shows how the annealing slows down for low internal energy. The whole process of tuning \tilde{v} has the physical meaning of **adaptation of the tolerance to the average distance of the particles to the target**.

Further into tunable parameters of the algorithm, we will want to **adapt the jump-covariance in parameter space to the empirical covariance of the population**: this can be achieved by introducing a scaling parameter β named **mixing speed**:

$$\Sigma_{jc}(\theta) \equiv \beta \Sigma_{emp} + \varepsilon \mathbb{1} = \beta \Sigma_{emp} + s \text{Tr}(\Sigma_{emp}) \mathbb{1} \quad (4.22)$$

Where Σ_{emp} is the covariance of the parameters θ_i taken from the population at a given step. Technically, this means preserving some memory (the quantity is set by β) about the system's condition while evaluating how to make the next jump proposal, so to look for the next θ s in the range in which the parameters are actually living at a given iteration. The last term proportional to the identity is instead needed to make sure the population does not get stuck on a linear submanifold (the logic behind it's similar of the Vhiola's algorithm).

Since making too small jumps does not ensure the endo-reversibility of the process, we must make sure that β is not too small w.r.t. \tilde{v} .

The last technique we can adopt is the addition of a **rejuvenation step every once in a while to kill trailing particles**. This is achieved by the *importance sampling* method, which we could sometimes exploit instead of the canonical updating schedule to set the temperature. Technically speaking, if the reversibility holds for every time, the population's pdf can be written in the form:

$$P_t(\mathbf{z}) = Z^{-1} f(\mathbf{z}) \exp\left(-\frac{u(\mathbf{z})}{T(1-\delta)}\right) \stackrel{1^{st} \text{ or}}{\simeq} Z^{-1} f(\mathbf{z}) \exp\left(-\frac{u(\mathbf{z})}{T}(1+\delta)\right)$$

where $f(\mathbf{z})$ is the joint prior and the term $1 - \delta$ (where δ is the **rejuvenation parameter** of the algorithm) ensures a low temperature value.

Having now a look at equation (4.14), we can notice how the idea behind rejuvenation consists of redefining the pdf by attaching to each particle \mathbf{z}_i a weight $w_i = \exp\left(-\delta \frac{u(\mathbf{z}_i)}{T}\right)$, namely $\mathbf{z}_i \mapsto (\mathbf{z}_i, w_i)$, and then perform the importance sampling according to this newly weighted data. By resampling from a multinomial distribution we can finally get new particles:

$$\mathbf{z}_i \xrightarrow{\text{rejuv}(\delta)} (\mathbf{z}_i, w_i) \xrightarrow{\text{resample}} (\tilde{\mathbf{z}}_i, 1) \equiv \tilde{\mathbf{z}}_i \quad (4.23)$$

The application of this method leads to **killing particles which are far from the target and duplicate particles close to the target**. Its benefit consists of narrowing the particles population towards the target, but it has the drawback of a loss in the effective sample size (i.e., we reduce the number of independent particles). This is the reason for which this last step must be not be used consistently are requires caution.

To sum up, the complete set of tuning parameters of the Simulated Annealing ABC (SABC) is given by the effective annealing speed \tilde{v} , the mixing speed β , and the rejuvenation parameter δ . The algorithm is quite robust w.r.t. these parameters. However, we must make sure that, depending on the annealing speed, the mixing is large enough, as otherwise we might violate the endo-reversibility assumption. This means that there might be additional entropy production within the system, or, in other words, that we might end up with an additional bias that is hard to control.

4.3 Summary statistics

Lesson 8 Considering a model $f(\mathbf{y}|\boldsymbol{\theta})$, summary statistics are functions from the high-dimensional model output \mathbf{y} to low-dimensional statistics $\mathbf{s}(\mathbf{y})$, that retain the more $\boldsymbol{\theta}$ -related information as possible

GC
AM

$$\mathbf{s} : \mathbf{y} \mapsto \mathbf{s}(\mathbf{y})$$

Definition 4.1 In particular, we say that summary statistics $\mathbf{s}(\mathbf{y})$ are **sufficient** if and only if they contain all the information about the parameters, i.e. if it is possible to write

$$f(\mathbf{y}|\boldsymbol{\theta}) = c(\mathbf{y})g(\mathbf{s}(\mathbf{y}), \boldsymbol{\theta}) \quad (4.24)$$

If we're doing Bayesian inference we can plug in the latter our observations

$$f(\mathbf{y}_{obs}|\boldsymbol{\theta}) = c(\mathbf{y}_{obs})g(\mathbf{s}(\mathbf{y}_{obs}), \boldsymbol{\theta}) \quad (4.25)$$

that leads to

$$\begin{aligned} f(\boldsymbol{\theta}|\mathbf{y}_{obs}) &\propto \int f(\mathbf{y}|\boldsymbol{\theta})f(\boldsymbol{\theta})\delta(\mathbf{y}-\mathbf{y}_{obs})d\mathbf{y} \propto \int f(\mathbf{s}|\boldsymbol{\theta})f(\boldsymbol{\theta})\delta(\mathbf{s}-\mathbf{s}(\mathbf{y}_{obs}))d\mathbf{s} \\ &\propto \int f(\mathbf{y}|\boldsymbol{\theta})f(\boldsymbol{\theta})d\Omega_{\mathbf{s}} \propto f(\boldsymbol{\theta}|\mathbf{s}(\mathbf{y}_{obs})) \end{aligned}$$

where $d\Omega_{\mathbf{s}}$ denotes the surface element induced by the Lebesgue-measure on the shell of constant $\mathbf{s}(\mathbf{y}_{obs}) = \mathbf{s}$.

■ **Example 4.4 — Ising model.** In the Ising model we can write the output distribution of spin configurations \mathbf{y} as

$$f(\mathbf{y}|\beta, h) = Z^{-1}(\beta, h) \exp \left(\beta \overbrace{\sum_{\langle i, j \rangle} y_i y_j}^{S_1(\mathbf{y})=\text{Energy}} + \beta h \underbrace{\sum_i y_i}_{S_2(\mathbf{y})=\text{Magnetization}} \right)$$

and it's manifest that the energy and the magnetization of a spin configuration are all we need to know in order to constrain the parameters (β, h) . ■

4.3.1 The exponential family

The Ising model belongs to a very special class of models, called **exponential family**. It encompasses all distributions with the functional form

$$f(\mathbf{y}|\boldsymbol{\theta}) = Z^{-1}(\boldsymbol{\theta})c(\mathbf{y}) \exp \left(- \sum_i g^i(\boldsymbol{\theta}) s_i(\mathbf{y}) \right) = Z^{-1}(\boldsymbol{\theta})c(\mathbf{y}) \exp(-\mathbf{g}(\boldsymbol{\theta})^T \mathbf{s}(\mathbf{y})), \quad (4.26)$$

that follows the structure of (4.25), therefore the $s_i(\mathbf{y})$ in the (4.26) are sufficient statistics.

In the case $g^i(\boldsymbol{\theta}) = \theta^i$ we call it a **natural parametrization** (it's not the case of the Ising model). The exponential family has the following property:

Theorem 4.2 — Pitmann-Koopmann-Dormois

Among the distributions whose domain does not depend on the parameters, only the family of exponential distributions has a finite number of sufficient summary statistics that does not grow unlimited with the number of degrees of freedom.

For example, in the Ising model, it doesn't matter how many spins there are, the summary statistics are always two.

As a counter-example we consider N i.i.d. samples following a Cauchy distribution, $y_i \sim f(y|\theta) \propto \frac{1}{1+\theta y^2}$, so that $f(\mathbf{y}|\theta) \propto \prod_i \frac{1}{1+\theta y_i^2}$.

Here you can't individuate any summary statistics of \mathbf{y} that has the form (4.25), hence you need every single y_i to retain all the information about θ . However, for practical purposes we don't really pretend to have every single bit of information, but we rather want to retain *almost all* of it. For example, given a set of \mathbf{y}_{obs} Cauchy-distributed draws, we can't infer θ by simply computing the first moments (as we would do for a gaussian distribution) because the Cauchy distribution has no converging moments. Nevertheless, its entropy is well-defined

$$S = - \int f(y|\theta) \ln f(y|\theta) dy = \ln(4\pi) - \frac{1}{2} \ln(\theta) \quad (4.27)$$

and we can retrieve from here almost all the info about θ . A more practical definition of sufficiency comes from information theory:

Definition 4.2 — Asymptotic sufficiency.

$s(\mathbf{Y})$ is asymptotically sufficient for Θ if and only if

$$I(\mathbf{S}, \Theta) \equiv I(s(\mathbf{Y}), \Theta) = I(\mathbf{Y}, \Theta) + \mathcal{O}(N^{-1}) \quad (4.28)$$

where N is the number of data points (spins for example) while I denotes the **mutual information** between correlated random variables defined as

$$I(\mathbf{S}, \Theta) := \int f(s, \theta) \ln \frac{f(s, \theta)}{f(s)f(\theta)} ds d\theta \quad (4.29)$$

Here, Θ is the parameter prior with given density $f(\theta)$ and, for given θ , $\mathbf{Y} \sim f(\mathbf{y}|\theta)$.

Notice that if s and θ are independent then the joint probability factorizes, and there's no mutual information.

Practical requirements for $s(\mathbf{y})$

The practical conditions that we wish to have from our summary statistics in ABC are

1. **Asymptotic sufficiency:** $I(\mathbf{S}, \Theta) = I(\mathbf{Y}, \Theta) + \mathcal{O}(N^{-1})$
2. **Asymptotic minimality:** of course the whole dataset is a sufficient statistics, but we want to have a minimality requirement. In information theory this translates in asking our summary statistics to have asymptotically a minimal entropy. This express our wish that as the number of degrees of freedom grows, noise should cancel out. In this way, our $s(\mathbf{y})$ retain as much information as possible about parameters but do not encode information about the noise.

We can express the mutual information as a difference of entropies:

$$\begin{aligned} I(\mathbf{S}, \Theta) &= \int f(s, \theta) \ln \frac{f(s, \theta)}{f(s)f(\theta)} ds d\theta = \int f(s, \theta) \ln \frac{f(s|\theta)f(\theta)}{f(s)f(\theta)} ds d\theta \\ &= - \underbrace{\int f(s, \theta) \ln f(s) ds d\theta}_{S(s)} + \underbrace{\int \frac{f(s, \theta)}{f(s)f(\theta)} \ln f(s|\theta) ds d\theta}_{f(s|\theta)f(\theta)} \\ &= S(s) - \underbrace{\int \left(- \int f(s|\theta) \ln f(s|\theta) ds \right) f(\theta) d\theta}_{S(s|\theta)} = S(s) - \langle S(s|\theta) \rangle_{\theta} \end{aligned}$$

Notice that this result is true for any couple of random variables. Since we want $I(\mathbf{S}, \Theta)$ to be maximum and $S(s)$ to be minimum then $\langle S(s|\theta) \rangle_{\theta}$ should go to 0. This means that conditions 1) and 2) lead to the so called **concentration property**

$$\langle S(s|\theta) \rangle_{\theta} = \mathcal{O}(N^{-1}) \rightarrow 0$$

This is an equivalent for general stochastic models of the well-known ensembles equivalence in statistical mechanics: as the number of degrees of freedom grows to infinity, all the fluctuations around the mean values of the thermodynamic variables (such as the energy or the number of particles) fade, and the result of the computations is the same either if take this variables fixed or if we allow them fluctuating around their average value. Hence, **we can interpret minimally sufficient statistics that satisfy 1) and 2) as generalized thermodynamic state variables.**

4.3.2 Phases and phase transitions in inference problems

From the concentration property we can get the following: sufficient summary statistics that are asymptotically sufficient and of minimal entropy will locally concentrate around a sub-manifold of the same dimension of the parameter's space of our model. We report here the example of a 3d summary statistics space and a model with 2 parameters. After fixing the parameters θ and retrieving some realizations $\mathbf{y} \sim f(\mathbf{y}|\theta)$, we calculate the correspondent $s(\mathbf{y})$, then, due to this concentration property, summary statistics will be concentrated on (potentially more than one) sub-manifolds of the dimension of the parameters space, in this case a 2d manifold. When we have more than one manifold, it is clear that we need an higher dimension for the summary statistics' space compared to the parameters' one to discriminate the different phases.

In statistical mechanics different phases emerge because of a competition between energy and entropy. When

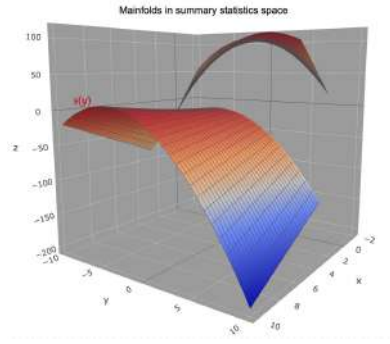


Figure 4.3: Summary statistics manifolds

these two are comparable, there's not a one-way equilibrium state for the system, but there could be a coexistence between two different phases: one which is energy-favoured and the other is entropy-favoured (think about the solid and the liquid phase of a liquid, respectively). This competition gives rise to a multi-modal free energy, each mode corresponding to a different phase:

$$F(s, \theta) = -\ln f(s, \theta) = -\ln \int \underbrace{f(\mathbf{y}|\theta)}_{\text{Energetic term}} \underbrace{d\Omega_s(\mathbf{y})}_{\text{Entropic term}}, \quad (4.30)$$

where $d\Omega_s(\mathbf{y})$ denotes the surface element induced by the Lebesgue-measure on the shell of constant $\mathbf{s}(\mathbf{y}) = \mathbf{s}$. If we have a large number of degrees of freedom that are integrated over in (4.30), the entropic term becomes important, and may lead to the emergence of thermodynamic states $\mathbf{s}(\mathbf{y})$ characterized by micro-states \mathbf{y} with a low probability/high energy, simply because there are so many micro-states associated with it (i.e. they come with a large phase-space volume $d\Omega_s(\mathbf{y})$). **For Bayesian inference, it is important to know which phase the observations belong to, because the posterior may strongly depend on it.**

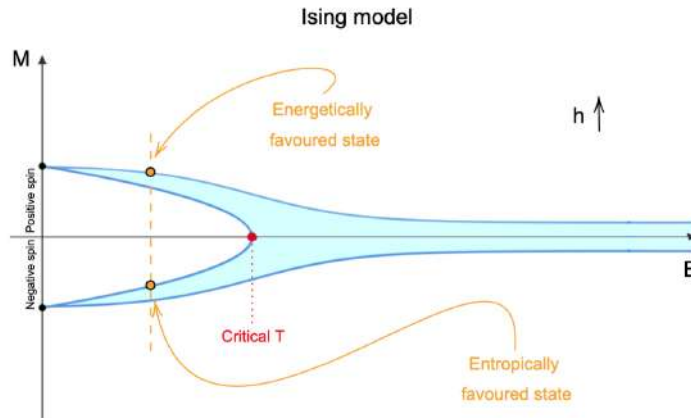


Figure 4.4: Distribution of summary statistics for the Ising model

■ **Example 4.5 — Ising model.** We report here, as an example, the distribution of summary statistics for the Ising model. We can see how, for $h = 0$, above a certain critical temperature there is no magnetization, whereas below both the spin-up and the spin-down magnetization are possible. If a small magnetic field is turned on, we can observe a two dimensional manifold embedded in the space. When we are in the ferromagnetic phase, we can either have a energetically favoured state where most of spins are aligned with h , or a entropically favoured state. In the latter case, we are a bit above the lower ceiling where all spins are anti-aligned with the external magnetic field, meaning that there are more ways for this macro-state to be realized than the energetically favoured state, even though each realization is less probable.

These two possible states could be inferred by the trend of the free energy, that displays two modes. Depending on the observed magnetization, the posterior will be skewed towards either one of the two states. ■

Stochastic non-linear model

We now move away from statistical mechanics to show that these concepts are important also to study stochastic models in general. We consider a stochastic non-linear model, in the form of an iterative map

$$y_{i+1} = \alpha g(y_i) + \sigma \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, 1)$$

In this model $\theta = (\alpha, \sigma)^T$, y_0 is given, and g is a non-linear function that admits two stable solutions, e.g. $x^2 e^{-x}$.

We start by studying the dynamics of this model forgetting the stochastic term: in this case the deterministic map will have two stable fixed-point, one for $y' = 0$ and the other at $y' > 0$ (see Figure 4.5). Depending on the value of α and on the curvature of g , the dynamics toward this second fixed point can be either monotonically converging, oscillating or even chaotic.

Now suppose that we turn on the noise. What is the fixed point corresponding to what we called *energetically favoured state*? Since the energy is the (negative) logarithm of a probability, the energetically favoured state will be the most probable one, namely the one corresponding to the deterministic solution ($\varepsilon = 0$). So, depending on the initial value of y_0 , it can be either y' or y'' . As the magnitude of the noise increases, the entropy of the system becomes more and more relevant, until the point in which the dynamics can converge toward the entropically favoured state. These two different model-behaviours correspond to different phases in statistical mechanics.

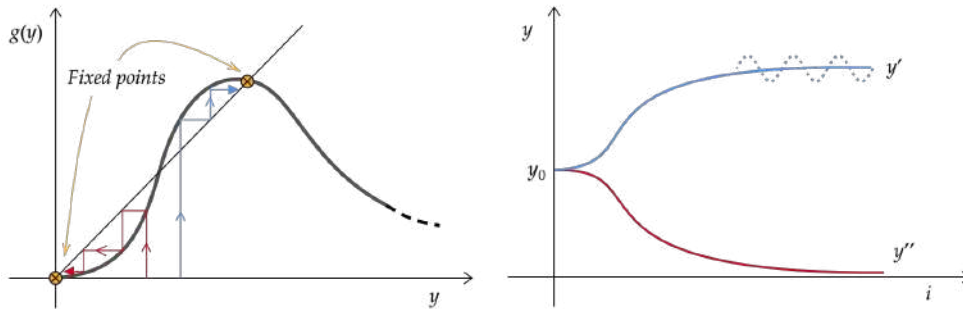


Figure 4.5: Deterministic dynamics toward one of the possible fixed points

Since we chose this model belonging to the exponential family, the number of its sufficient summary statistics is bounded and we can easily parametrize our time-series likelihood $f(\mathbf{y}|\theta)$ like follows:

$$\begin{aligned} f(\mathbf{y}|\theta) &\propto \exp \left[-\frac{1}{2\sigma^2} \sum_i (y_{i+1} - \alpha g(y_i))^2 \right] \\ &= \exp \left[-\frac{1}{2\sigma^2} \underbrace{\sum_i y_{i+1}^2}_{\tilde{s}_1(\mathbf{y})} + \frac{\alpha}{\sigma^2} \underbrace{\sum_i y_{i+1} g(y_i)}_{\tilde{s}_2(\mathbf{y})} - \frac{\alpha^2}{2\sigma^2} \underbrace{\sum_i g(y_i)^2}_{\tilde{s}_3(\mathbf{y})} \right] \end{aligned} \quad (4.31)$$

Here we need three different summary statistics even if $\dim(\theta) = 2$, because there are three linearly independent functions of the parameters in front of them. This can seem quite surprising, since we usually need a number of summary statistics equal to the number of parameters. Indeed, this is actually true unless we are in presence of phase transitions: in this case this third summary statistics has the role of an order parameter telling us what is the phase we ended up. To see this, let's redefine the summary statistics through

the bijective function $\mathbf{S}(\mathbf{s})$:

$$S_1(\mathbf{y}) = \hat{\alpha}(\mathbf{y}) = \frac{\sum_{i=0}^{N-1} y_{i+1} g(y_i)}{\sum_i g(y_i)^2} \quad \text{auto-correlation estimator}$$

$$S_2(\mathbf{y}) = \hat{\sigma}(\mathbf{y}) = \frac{1}{N} \sum_{i=0}^{N-1} (y_{i+1} - \hat{\alpha}(\mathbf{y}) g(y_i))^2 \quad \text{sigmas/noise estimator}$$

$$S_3(\mathbf{y}) = \frac{1}{N} \sum_{i=0}^{N-1} g(y_i)^2 \quad \text{order parameter}$$

Here $S_1(\mathbf{y}), S_2(\mathbf{y})$ are actually MLE parameter-estimators regressors that can be used for estimating α and σ , but they don't tell anything about the phase. To tell in which phase we are, we need to compute $S_3(\mathbf{y})$: when we are in y'' all the g_i will be close to 0, whereas when g_i display higher values we are in the "upper" phase y' . When we do Bayesian inference it's important to have also this information, because the two posteriors can be very different and contain different amount of information (such as the average value of y around which the dynamic happens, which is also important for inferring α).

Infer summary statistics: Machine Learning

When the summary statistics of the problem we are studying are not known, they can be retrieved for example through machine learning regression techniques $\mathbf{y} \xrightarrow{\text{ML}} \mathbf{s}(\mathbf{y})$. After sampling $\theta_i \sim f(\theta)$, it is possible to sample realizations $\mathbf{y} \sim f(\mathbf{y}|\theta)$ and then, through the chosen ML technique, we try to infer $\hat{\theta}$ from \mathbf{y} through regression: this will be our $s(\mathbf{y})$

$$\theta_i \sim f(\theta) \rightarrow \mathbf{y} \sim f(\mathbf{y}|\theta) \quad (4.32)$$

$$s(\mathbf{y}) := \hat{\theta}(\mathbf{y}) \xleftarrow{\text{lin reg}} \mathbf{y} \quad (4.33)$$

However, this regression models are limited by the fact that they give you just as many summary statistics as the number of parameters involved. To work-around this problem people came out using autoencoders. Starting from a high-dimensional variable \mathbf{y} , these models are able to compress it in $\mathbf{s}(\mathbf{y})$ (also called *bottleneck*) and then they try to reconstruct the input. This way the autoencoder just retain information about parameters and cuts-off the noise.

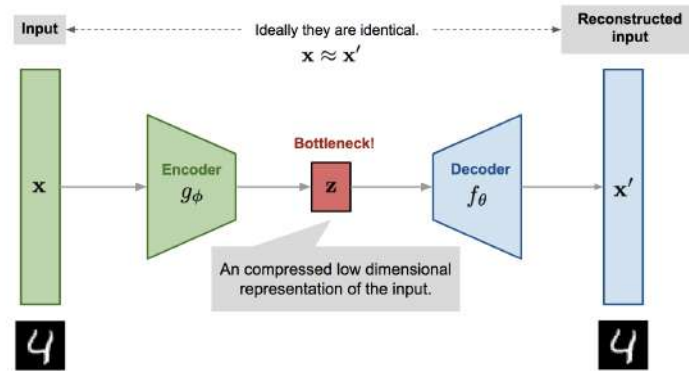
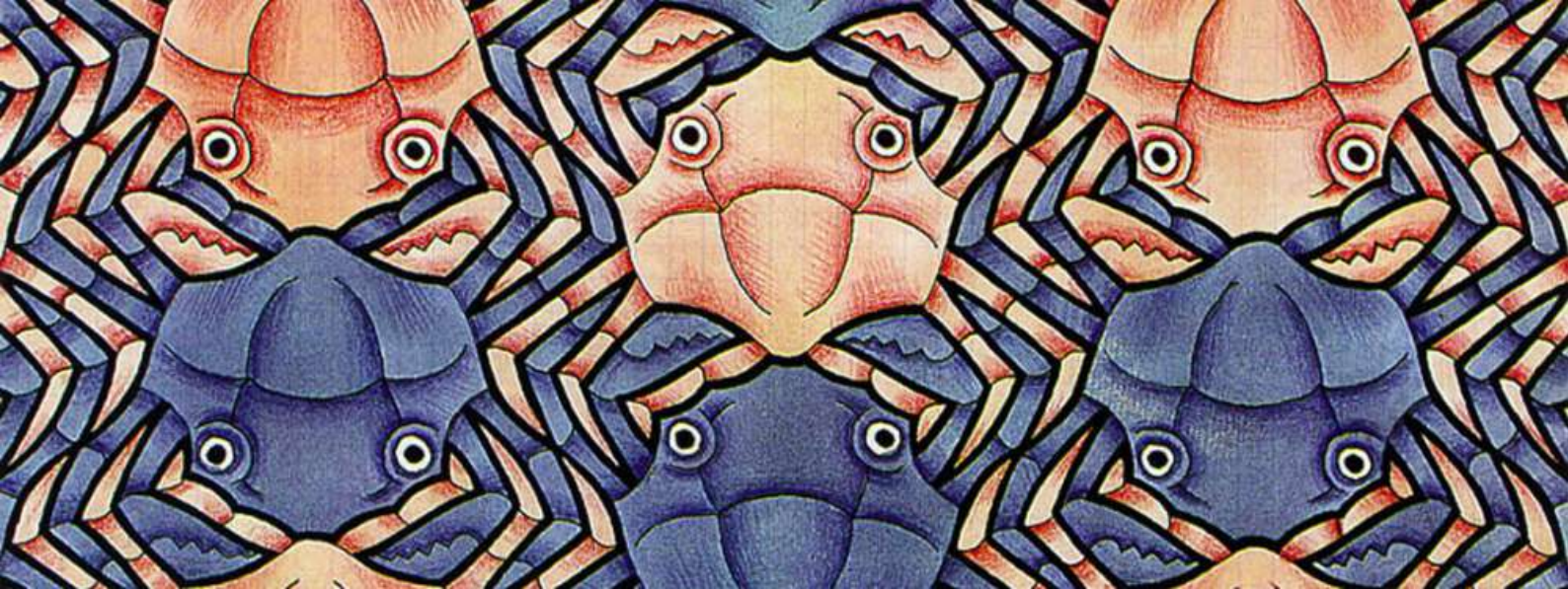


Figure 4.6: Schematic representation of an autoencoder



5. ML-approaches to Bayesian Inference

5.1 Introduction

So far, methods we introduced were especially meant for sampling from the posterior: $\theta_i \sim f(\theta|y_{obs})$. Metropolis, HMC and ABC (if sufficient summary statistics are used) are all asymptotically exact, i.e. in the limit of infinite number of iterations $N \rightarrow \infty$ parameters are sampled from the exact posterior, but are also very expensive.

Lesson 11
26/04
AM

To overcome this issue, we could exploit Machine Learning (ML) techniques by turning Bayesian inference into an optimization problem.

Definition 5.1 — Neural density estimators. We define neural density estimators the probability distributions $q_\phi(\theta)$ belonging to a large parametrized family that determines the hypotheses class for the ML optimization.

The goal of ML will be finding the optimal set of parameters ϕ^* so to find the neural density estimator which is the closest to the given pdf (prior, posterior or likelihood): $q_{\phi^*} \approx f$. The distance between the functions can be measured, for example, as the Kullback–Leibler (KL) divergence and is the loss function of the ML model.

The type of models used for these density estimators are neural network models, which make it easy to calculate derivatives of the loss function w.r.t. ϕ (via backpropagation, which essentially is the chain rule), and therefore allow optimization through steepest descent methods. For Bayesian inference, we are thus using (conditional) neural density estimators.

Even if the optimization is efficient, it can lead to **biased results**, depending on how large the hypotheses class (i.e., the set of models between which we are looking for the best estimator) is. Therefore, one may want to find an optimal threshold between accuracy and efficiency.

5.2 The variational Bayes method

A popular way to estimate via neural network the posterior for a given model is given by the variational inference (or variational Bayes) method. Since the KL-divergence can be used as a "distance of functions" to estimate closeness between target and best guess, in this method the loss function is set to be the

KL-divergence in the form:

$$\begin{aligned} KL(q_\phi || f_{post}) &= \int q_\phi(\theta) \ln \left(\frac{q_\phi(\theta)}{f(\theta | \mathbf{y}_{obs})} \right) d\theta = \int q_\phi(\theta) \ln \left(\frac{q_\phi(\theta) f(\mathbf{y}_{obs})}{f(\mathbf{y}_{obs} | \theta) f(\theta)} \right) d\theta \\ &= \ln f(\mathbf{y}_{obs}) \underbrace{\int q_\phi(\theta) d\theta}_{=1} + \int q_\phi(\theta) \ln \left(\frac{q_\phi(\theta)}{f(\theta)} \right) d\theta - \int q_\phi(\theta) \ln f(\mathbf{y}_{obs} | \theta) d\theta = \end{aligned} \quad (5.1)$$

$$= \ln f(\mathbf{y}_{obs}) + \mathcal{F}(\phi) \quad (5.2)$$

where we've defined

$$\mathcal{F}(\phi) = \underbrace{\int q_\phi(\theta) \ln \left(\frac{q_\phi(\theta)}{f(\theta)} \right) d\theta}_{-S} - \underbrace{\int q_\phi(\theta) \ln f(\mathbf{y}_{obs} | \theta) d\theta}_{-U} \quad (5.3)$$

The term $\mathcal{F}(\phi)$ is sometimes called "**variational free energy**", since it can be seen as $\mathcal{F}(\phi) = U - S$ given the fact that the first term of equation (5.3) is equal in form to a (-) relative entropy, and the expectation value over θ of the negative logarithm of a pdf can be seen as an energy term (please notice that those are only conventional terms). Another name for it is "**negative evidence lower bound**" (- ELBO), since in the limit $KL(q_\phi || f_{post}) \rightarrow 0$ it returns, with a minus sign, a bound of the evidence $f(\mathbf{y}_{obs})$.

Looking at equation (5.2) is clear that since $f(\mathbf{y}_{obs})$ does not depend on ϕ , the ML algorithm will find the best parameters by minimizing the variational free energy $\mathcal{F}(\phi)$.

But what is commonly a good choice for the family of neural density estimators? The best way to tackle problems like the one above without wasting too many computational resources is to work on transformations instead on single models. That is, we agree on a default, simple basis model q_0 (e.g., the normal Gaussian) and define a set of transformations $\Phi_\phi : \Theta \rightarrow \Theta$ such that we can express any estimator q_ϕ in terms of Φ and q_0 , as shown in the following equations. Firstly, we must recall that pdfs do not transform as functions but as densities and thus we must preserve the volume by taking into account the Jacobean on the transformation:

$$\text{In 1D, let } y = \Phi(x) \text{ and } f(y) dy = f(x) dx, \text{ then } f(y) = f(\Phi^{-1}(y)) \frac{dx}{dy} \text{ and } dy = \frac{d\Phi(x)}{dx} dx \quad (5.4)$$

So, in our case:

$$q_\phi = q_0(\Phi_\phi^{-1}(\theta)) \cdot \left(\det \left| \frac{d\Phi_\phi(\theta)}{d\theta} \right| \right)^{-1} \quad (5.5)$$

Hence, rewriting equation (5.3) as

$$\begin{aligned} \mathcal{F}(\phi) &= \left\langle \ln \frac{q_\phi(\theta)}{f(\theta)} - \ln f(\mathbf{y}_{obs} | \theta) \right\rangle_{q_\phi} = \\ &= \langle \ln q_\phi(\theta) - \ln f(\mathbf{y}_{obs}, \theta) \rangle_{q_\phi} + \text{const. independent of } \phi \end{aligned}$$

and recalling that equation (5.4) formally means that expected values are preserved under transformation of parameters, we conclude:

$$\begin{aligned} \mathcal{F}(\phi) &= \langle \ln q_\phi(\theta) - \ln f(\mathbf{y}_{obs}, \theta) \rangle_{q_\phi} = \\ &= \langle \ln q_\phi(\Phi_\phi(\theta)) - \ln f(\mathbf{y}_{obs}, \Phi_\phi(\theta)) \rangle_{q_0} = \\ &= \left\langle \ln q_0(\theta) - \ln \det \left(\frac{d\Phi_\phi(\theta)}{d\theta} \right) - \ln f(\mathbf{y}_{obs}, \Phi_\phi(\theta)) \right\rangle_{q_0} \end{aligned} \quad (5.6)$$

In particular, the last expression above shows how the expected value can be evaluated directly on q_0 (chosen *a priori*) and does not depend on ϕ . As a result, we can calculate it in a Monte Carlo fashion:

$$\mathcal{F}(\phi) \approx \frac{1}{N} \sum_{i=1}^N \left[-\ln \det \left(\frac{d\Phi_\phi(\theta_i)}{d\theta} \right) - \ln f(\mathbf{y}_{obs}, \Phi_\phi(\theta_i)) \right] + C(\mathbf{y}, \theta) \quad (5.7)$$

Where $\theta_i \sim q_0(\theta)$. The goal of the ML algorithm reduces thus to **minimize the sum of two terms** (since the last does not depend on ϕ). To accomplish this task in a good way, the set of transformations $\Phi(\theta)$ must belong to a class of functions that satisfy two conditions:

1. Must be sufficiently flexible and expressive (in order to model any distribution that we would like);
2. Must be invertible (and, since calculating the Jacobean scales badly w.r.t the number of parameters, should allow also an easy computation of the inverse and their derivatives w.r.t ϕ).

Proven this condition as satisfied, then we can use it both for calculate density functions as done in Eq. 5.5 and to sample data, i.e.

$$\theta_i \sim q_0(\theta) \iff \Phi_{\phi_i}(\theta) \sim q_\phi(\theta)$$

A popular set of transformations respecting such properties is the one of **auto-regressive (AR) transformations**.

Definition 5.2 — Auto-regressive transformations. Given a set of parameters $\theta = (\theta_1, \theta_2, \dots, \theta_D)$, a transformation $\Phi_{\phi,i}(\theta)$ with index i is said to be auto-regressive if it is a bijective function of the i -th parameter θ_i and of an arbitrary complex function of all the previous parameters $\theta_j, j < i$:

$$\Phi_{\phi,i}(\theta) = h(\theta_i; \psi_i(\theta_1, \dots, \theta_{i-1})) \quad (5.8)$$

For the specific case of ML applications, the functions $\psi_i(\theta_{j < i})$ will be neural networks.

A function of this form trivially satisfies invertibility (since h is bijective) and has a cheap determinant (it looks clear that $d\Phi_i/d\theta_j = 0$ for $j > i$, hence the Jacobean is a triangular matrix and thus the determinant is just the product of diagonal terms), allowing a gain in scalability from $\mathcal{O}(N_{param}^3)$ to $\mathcal{O}(N_{param})$. Moreover, **a universality theorem states that they can approximate any suitably regular distribution with arbitrary accuracy.**

In practise, it is usual to pick as ψ_i some simple models: a common choice is the **linear auto-regressive model**

$$h(\theta) = \psi_1 \theta + \psi_2 \text{ where } \psi_i : \mathbb{R}^{i-1} \rightarrow \mathbb{R}^2 \ni (\psi_1, \psi_2) \quad (5.9)$$

Which is the standard affine model employed in neural networks, up to the activation function which can be decided independently. However, the simplicity of this method is usually too much to ensure universality: hence, it is a good practice to stack many AR-transformations on the top of each other and exploit the chain rule for the backpropagation phase in NNs. This technique is known as "**auto-regressive flow**":

$$q_0 \quad \theta \xrightarrow{\Phi_{\phi_1}} \theta^{(1)} \xrightarrow{\Phi_{\phi_2}} \dots \xrightarrow{\Phi_{\phi_k}} \theta^{(k)} \quad (q_\phi)$$

We refer to auto-regressive flow with $q_0 = \mathcal{N}(0, 1)$ in terms of "normalized flow" (since moving backwards until the zero stage we ultimately find a normal distribution). More details can be found in [KPB20].

5.3 ML alternatives to ABC - I

The method above requires many evaluations of the likelihood function and its derivatives (like in HMC). When those calculations are too computationally expensive, it is convenient to exploit other methods such as ABC [PSM19]. Here we present the first method to enhance algorithm performances by **learning the posterior via ML**. In this case, the KL-divergence representing the loss function is:

$$KL(f_{post} || q_\phi) = \int f(\theta | y_{obs}) \ln \left(\frac{f(\theta | y_{obs})}{q_\phi(\theta | y_{obs})} \right) d\theta = - \langle \ln q_\phi(\theta | y_{obs}) \rangle_{f(\theta | y_{obs})} + C(y_{obs}, \theta) \quad (5.10)$$

Where the C term is independent of ϕ . We observe that this function is the complementary of the previous algorithm, that is, the KL-divergence in which the roles of the posterior and the neural density estimator are inverted.

Focusing on the task, we can thus neglect the C term and look for the best set of parameters to maximize the expected value of the logarithm of the estimator:

$$\min_{\phi} \left(KL(f_{post} || q_\phi) \right) \iff \max_{\phi} \left(\langle \ln q_\phi(\theta | y_{obs}) \rangle_{f(\theta | y_{obs})} \right) \approx \max_{\phi} \left(\frac{1}{N} \sum_{i=1}^N \ln q_\phi(\theta_i | y_i) \right) \quad (5.11)$$

Where the approximation is made in a Monte Carlo fashion with $\theta_i \sim f(\theta)$ and $y_i \sim f(y | \theta_i)$. Please notice that this last expression does not exploit the observations $\{y_{obs}\}$, thus the method could be highly inefficient:

we must pay attention while sampling \mathbf{y} so to have these values sufficiently "close" to the observed data w.r.t the set of parameters ϕ .

It is important to highlight how in this case q_ϕ has an explicit dependence on \mathbf{y}_{obs} : from a practical point of view, we could see the ML parameters ϕ as the weights of a neural network mapping from the space of outputs \mathbf{Y} to the space of some parameters, $\Xi(\mathbf{y}, \phi)$, which parametrize the space of approximate posteriors q_ϕ .

In a nutshell: Since these “parameters” start flourishing everywhere, let’s make a recap to avoid getting confused:

- θ , as usual, are the parameters of the model we want to calibrate, i.e. that we sample from the posterior once we’ve learnt it;
- ϕ are the “structural” parameters of the ML machinery that we tune with the training procedure. For example, they can be thought as the weights of a neural network;
- The neural density estimator depends on ϕ and \mathbf{y} through Ξ , which can be seen as a set of “high level” parameters coming out from a neural network (see Figure 5.1)

■ **Example 5.1** An expressive example of the case we are studying is to consider as neural density estimator family the set of Gaussian mixtures in the form:

$$q_\phi(\theta|\mathbf{y}) = \sum_{k=1}^K \omega_{\phi,k}(\mathbf{y}) \mathcal{N}(\mu_{\phi,k}(\mathbf{y}), \Sigma_{\phi,k}(\mathbf{y}))$$

Where in this case $\Xi \equiv [(\omega_{\phi,1}, \mu_{\phi,1}, \Sigma_{\phi,1}), \dots, (\omega_{\phi,K}, \mu_{\phi,K}, \Sigma_{\phi,K})]$. In order to find the best approximated posterior we could proceed in two ways:

1. Implement a neural network which takes outputs \mathbf{y} as inputs and returns the optimal parameters Ξ for the estimator (in this case the set ϕ represents the set of weights on the edges of the network);
2. Adopt the previously mentioned AR flow technique and define the mode transformations:

$$\Phi = \Phi_{\phi,i}(\theta, \mathbf{y}) = h(\theta_i, \psi(\theta_{j<i}, \mathbf{y})) \quad (5.12)$$

For this particular case, it can be shown that the latter method is better performing. ■

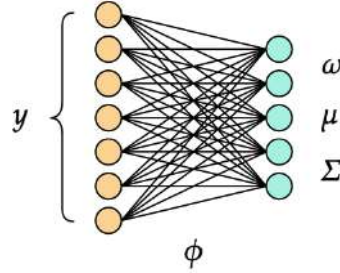


Figure 5.1: Ξ estimation trough a neural network

As mentioned above, this method allows to get approximate posteriors for any dataset \mathbf{y} and its quality for a specific task is strictly bounded by the ability to focus on some given observations \mathbf{y}_{obs} . Hence, a suitable way to stay close to the observations would be perform **importance sampling in an iterative way**, starting from an original prior $f(\theta)$ and replacing it with a proposal prior $\tilde{f}(\theta)$ which is updated every time setting it equal to the approximate posterior which the ML algorithm returns.

Algorithm 5.1 — Iterative importance sampling for ML-ABC I Consider the loop of two steps:

- **STEP 1:**
 - Set the first proposal prior $\tilde{f}(\theta) = f(\theta)$;
 - Sample $\theta_i \sim \tilde{f}(\theta)$ and $\mathbf{y}_i \sim f(\mathbf{y}|\theta_i)$;
 - Train $q_\phi(\theta|\mathbf{y})$ so to obtain the best set of parameters ϕ^* according to equation (5.11);
 - Set the new proposal prior $\tilde{f}(\theta) = q_{\phi^*}(\theta|\mathbf{y}_{obs})$;

• **STEP 2:**

- Sample $\theta_i \sim \tilde{f}(\theta)$ and $y_i \sim f(y|\theta_i)$;
- Train $q_\phi(\theta|y)$ so to obtain the best set of parameters ϕ^* ;
- Set as new approximate posterior

$$q_\phi^{new}(\theta) = q_{\phi^*}(\theta|y_{obs}) \frac{f(\theta)}{\tilde{f}(\theta)}$$

by importance sampling (the last term acts as a correction that quantifies the approximation made by sampling from \tilde{f} instead of the true prior f);

- Set the new proposal prior as the best neural density estimator of the posterior so far: $\tilde{f}_{new}(\theta) = q_\phi^{new}(\theta)$.

This way the result gets closer at each loop to the true posterior. Iterate STEP 2 until the result stabilizes.

★

5.4 ML alternatives to ABC - II

While the previous algorithm focuses on learning the posterior, an alternative way to use learning algorithms to enhance ABC is to **learn the conditional probability density** $f(y|\theta) \simeq q_{\phi^*}(y|\theta)$, i.e., find the best parameters for a "conditional density estimator" $q_\phi(y|\theta)$. Then, we can use standard methods to sample from a posterior obtained as the product of the prior and the learned likelihood.

Like before, we must specify the objective function which is again the KL-divergence in the form:

$$KL(f_{likeh}||q_\phi) = \int f(y|\theta) \ln \left(\frac{f(y|\theta)}{q_\phi(y|\theta)} \right) dy = -\langle \ln q_\phi(y|\theta) \rangle_{f(y|\theta)} + C(y, \theta) \quad (5.13)$$

Like for the previous algorithm minimizing the KL-divergence by modifying ϕ corresponds to maximize the approximate expected value of the logarithm of the density estimator:

$$\min_{\phi} \left(KL(f_{likeh}||q_\phi) \right) \iff \max_{\phi} \left(\langle \ln q_\phi(y|\theta) \rangle_{f(y|\theta)} \right) \approx \max_{\phi} \left(\frac{1}{N} \sum_{i=1}^N \ln q_\phi(y_i|\theta_i) \right) \quad (5.14)$$

Where $\theta_i \sim \tilde{f}(\theta)$ is drawn from a proposal prior and $y_i \sim f(y|\theta_i)$ are the model simulations.

In this case the choice of the prior does not really affect the results: in fact, the neural network only focuses on understanding how samples y behave given the parameters θ . Nevertheless, like before a choice for the proposal prior which is close to the true posterior and that depends on the observed data y_{obs} will be beneficial for the simulation.

Algorithm 5.2 — Iterative importance sampling for ML-ABC II Consider the loop of two steps:

• **STEP 1:**

- Set the first proposal prior $\tilde{f}(\theta) = f(\theta)$;
- Sample $\theta_i \sim \tilde{f}(\theta)$ and $y_i \sim f(y|\theta_i)$;
- Introduce an archive to store all the previously drawn params and samples: $\mathcal{A} = \{(y_i, \theta_i)\}$;
- Train $q_\phi(y|\theta)$ on \mathcal{A} so to obtain the best set of parameters ϕ^* according to equation (5.14);
- Get the approximate posterior $\tilde{f}(\theta|y_{obs})$ as the product between the true prior and the conditional density estimator (approximated likelihood), which will be eventually a proportional approximation of the true posterior: $\tilde{f}(\theta|y_{obs}) = q_{\phi^*}(y_{obs}|\theta) \cdot f(\theta) \propto f(\theta|y_{obs})$.

So far the representation is still poor (the true prior is still much broader than the true posterior), but the second step improves the results.

• **STEP 2:**

- Set as new proposal prior the approximated posterior of step 1: $\tilde{f}_{new}(\theta) = \tilde{f}(\theta|y_{obs})$;
- Sample $\theta_i \sim \tilde{f}_{new}(\theta)$ and $y_i \sim f(y|\theta_i)$ (e.g., via MCMC). Since we only want to train the model for y_i conditioned to θ_i , add the new samples (y_i, θ_i) to \mathcal{A} ;
- Train $q_\phi(y|\theta)$ on the new archive \mathcal{A} so to obtain the best set of parameters ϕ^* ;
- Get the approximate posterior as in step 1: $\tilde{f}(\theta|y_{obs}) = q_{\phi^*}(y_{obs}|\theta) \cdot f(\theta)$.

At the end of the second step the approximate posterior should be closer to its true form. The last step can be iterate until a satisfactory result is achieved. ★

In this algorithm sampling is done iteratively, starting with the prior and then narrowing in more and more on the posterior. The advantage of learning the output density is that all the output samples generated in previous iterations can be used as well (the “wrong” prior does not matter because we are conditioning on θ). This is the reason to introduce an archive and store all generated samples to enlarge the training set for the neural network.

Finally, it is also worth mentioning that also for the conditional probability estimation AR-flows are possible solutions.

Part II

Jeff Byers



6. The Bayesian approach

6.1 On the Bayesian interpretation

6.1.1 Physics as Encoding, Decoding and Bottlenecks

Information theory is about compressing information down, and find what matters inside a larger set of data. Have a look at figure 6.1. The y object on the left could be the result of an experiment, or ourselves out in the world collecting data. Since this huge amount of information doesn't fit inside our brain, we need to compress it down into something that generalize it and keeps only the important features. This "something" is what we call a **model**, and the x in the figure can be thought as its parameters. This compression process is nothing but the *inference* process. Eventually, we want to use our calibrated model to predict the future, and conceptually this is like *decoding* our compressed information into simulated data. Ideally, a good inference process has been accomplished if the input and the reconstructed one match together.

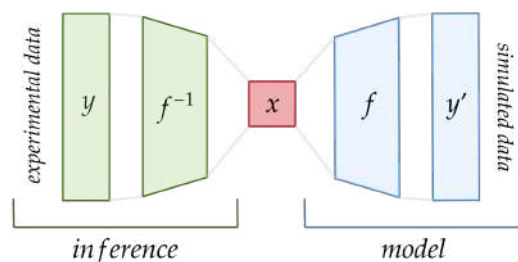


Figure 6.1: Encoding and decoding world data

The Bayesian inference is on the left. There are several types of model that we can choose for representing the process underneath our data. By looking at figure 6.2, those models that are typical of physics are in the upper left corner. In natural sciences we often deal with **causality**, a massive amount of **theory** is supporting us and there are just a few **parameters** that describe systems.

The opposite of physics is distinguishing images from the net of cats and dogs. There is nothing deep underlying images of animals, we just have an enormous amount of data, and we have to learn features with statistical correlations in order to be able to predict fresh images of cats and dogs in the future. The problem tends to be **non-parametric** since we don't know a priori how many features we will need or how many of them our unsupervised algorithm will be able to find out.

Classical statistics is an intermediate case: there is some theory and formulas to calculate, say, a standard deviation or a Gaussian probability density function, but that is enough to describe the model.

The upper right part, where there is causality and no parameters are predefined, is kind of a terra incognita. We will discuss this part also taking into account the rightmost part of the table, where we see **explanation**

Lesson 9
19/04
FC
TF
Lesson 10
23/04
LR

and **predictions**: people in machine learning sometimes are happy to be able to predict correctly but they often do not know how this happens. But when there is the need of *interpretability*, machine learning is not enough, and we need to build different kind of models.

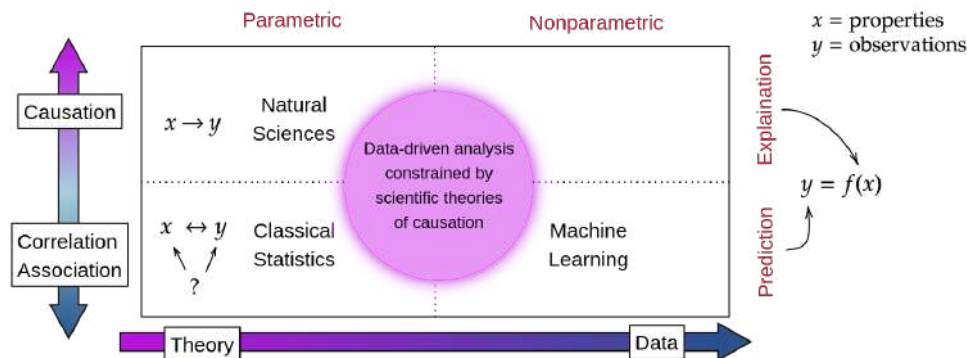


Figure 6.2: Types of models in Bayesian inference

6.1.2 The Bayes' Theorem

It's worth to spend some time looking deeper at the Bayes' formula

$$\underbrace{P(\text{model} \mid \text{data})}_{\text{Posterior}} = \frac{\underbrace{P(\text{data} \mid \text{model})}_{\text{Likelihood}} \underbrace{P(\text{model})}_{\text{Prior}}}{\underbrace{P(\text{data})}_{\text{Evidence}}}$$

The interpretation is the following. We collected some data, and we propose some possible models which are supposed to describe more or less the particular occurrence we got. For concreteness, suppose we assume a Gaussian distribution for the data and we want to infer the parameter μ (however, we could also have totally different models); in this case, when we say “model”, we’re referring to a particular value of μ . The probability that we assign to each of those values is given by two contributions.

The first one is the **Prior**, namely the knowledge, or belief, that we have on a particular model, and it somehow express our ignorance about the problem. Remember that learning is a process, and the prior is actually where we start from.

The second term carries the information we have from our data, and it is the **likelihood**. An important remark has to be done about the likelihood: it is *not* a probability distribution. When we write $P(\text{data} \mid \text{model})$, we have to bare in mind that the **data are fixed**, while the particular model is not. This means that the likelihood is a **function of the model (or parameters) given the data**. For example: suppose you made a survey, and you discovered that 30 out of 150 people prefer the color brown to the color red. Then, if x is the number of people who prefer brown, the likelihood will be $\text{Binom}(x = 30 \mid N = 150, p)$, and it is a function of p , not a pdf!

It is also a dangerous practice to choose the model by evaluating only the likelihood (Maximum Likelihood Estimation, MLE), because this tend to **overfit** the data. In the Bayesian context, this cannot happen because there is something that regularizes the choice of models, and this is the prior.

The theoretical way of thinking about the world, is that we have the answer (which is like a model) and our data is noisy, around the model. The Bayes theorem is telling us to think in a different way: you measured something, the data is fixed, and the model is **fluctuating around the data**. Maybe our data is noisy, but compared to what?

The Bayes formula can be used also for comparing different models. The idea is that our data carry a fixed amount of probability (the **evidence** at denominator) and we have to spread it among all the different models. This means that if we pretend to describe the phenomenon with a complicated model having a lot of parameters, each particular set of those values will have a small probability. On the other hand, a simple model with a couple of parameters will have a higher probability for each particular choice. This can be seen as a sort of Occam’s razor principle that keeps us safe from overfitting the data.

6.1.3 What is probability?

How can we define probability, and how can we interpret it? According to MacKay, we should start from the concept of **ensemble**.

Definition 6.1 — Ensemble. An ensemble X is a triple $(x, \mathcal{A}_X, \mathcal{P}_X)$, where the outcome x is the value of a random variable, which takes on one of a set of possible values, $\mathcal{A}_X = \{a_1, a_2, \dots, a_i, \dots, a_I\}$, having probabilities $\mathcal{P}_X = \{p_1, p_2, \dots, p_I\}$, with $P(x = a_i) = p_i$, $p_i \geq 0$ and $\sum_{a_i \in \mathcal{A}_X} P(x = a_i) = 1$.

Figure 6.3 is a nice representation of this concept. All the possible outcomes from the phenomenon we're measuring represent the **Sample space**. Each time we do a measure, we get an **outcome** x . The set of all the propositions we can make about the outcome is called **Event space**, and possibly it can contain a huge number of elements. For example, a proposition can be: “I get an outcome that can be either 2 or 3”. Finally, we map each proposition into a real number, the **probability**, which can be seen as the corresponding volume in the event space.

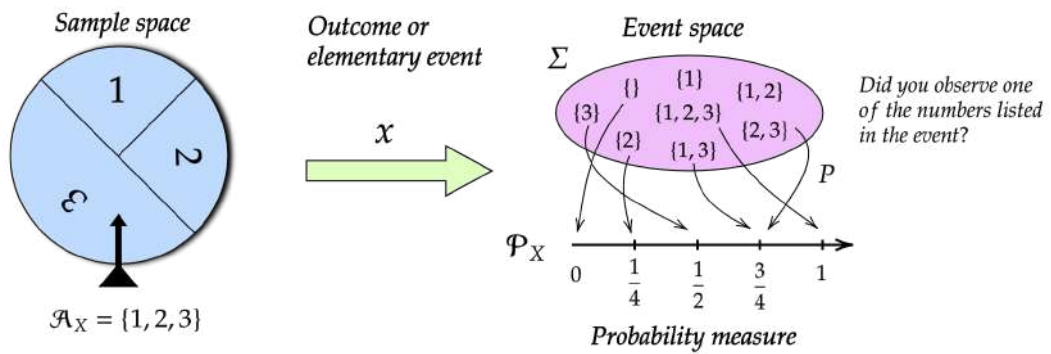


Figure 6.3: Ensemble

Now comes a delicate question: does the probability belong to *the world*, or is it in *the model*? The answer is not trivial at all, especially for physicists who are trained to think that all the predictions we can make about the fundamental processes of Nature are only probabilistic. Really nice reasoning about this point can be found in [Jay89], and we strongly suggest you to have a look at it. Essentially, in the first part the author retrieves Einstein's formula for the diffusion coefficient through Bayesian reasoning, and it goes as follows.

Bayesian inference in diffusion processes

Consider a diffusion process, and denote with $n(x, t)$ the density of particles at position x and time t . We know (phenomenologically) that the flux of particles is $J = -D \nabla n$, and we want to compute the diffusion constant D . Starting from position x at time t , from the central limit theorem we can assert that the probability for the particle to be at position y at time $t + \tau$ is given by a gaussian distribution

$$P(y|x, I) = A \exp \left[-\frac{(y-x)^2}{2\sigma^2(\tau)} \right].$$

However, there's no way to predict “ab initio” a flux of particles from there, because the distribution is symmetric and the average displacement is zero. The solution is to think the other way around: *what is the probability that the particle was at position z , given that now it is at position x ?* What makes all the difference here, is that by focusing on the past we have additional information: the equations of motions are symmetric in past and future, but our information about the particles is not. Hence, the *posterior* probability for the particle to be at position z is

$$P(z|x, t, I) = A P(x|z, I) P(z|I).$$

Now, of course the prior probability to be at position z must be proportional to the particles' density at that point. By taking the logarithm of both sides and replacing all the pieces we get

$$\ln P(z|x, I) = \ln n(z) - \frac{(z-x)^2}{2\sigma^2(\tau)} + \text{const.}$$

Differentiating, we find that the most probable value of the past position z is not x , but

$$\hat{z} = x + \sigma^2 \nabla(\ln n) = x + (\delta x)^2 \nabla(\ln n).$$

The average flux over a time interval 2τ centered in the present is

$$J = n \bar{v} = n \frac{x(t + \tau) - x(t - \tau)}{2\tau},$$

and replacing $x(t + \tau) \equiv x$, $x(t - \tau) \equiv z$, we get

$$J(x, t) = -\frac{(\delta x)^2}{2\tau} \nabla n \implies D = \frac{(\delta x)^2}{2\tau}.$$

Notice that we didn't even assume that the gradient of n must be present: Bayes' theorem told us that automatically. What can seem strange, here, is that we updated the propagator of motion with our prior information about the process; a physicist would say that we're not allowed to modify the laws of Nature merely because our state of knowledge has changed! The point here is that the gaussian distribution above (and probability in general) *is not* a law of Nature: it's just a way we have to model our ignorance about the world. As such, it's perfectly reasonable to update it as soon as our state of knowledge changes.

In a nutshell: Probability quantifies uncertainty in our representations and the corresponding inference procedures from data to models. Probability is just a tool in order to understand what is going on, it **is not** a physical property! It's just a "map" from *us* to the world that help us to describe it, but it's not actually *in the world*. To appreciate the distinction between physical predictions and inference it is essential to recognize that propositions at two different levels are involved, namely the *ontological* level (the laws of Nature) and the *epistemological* level (concerning the knowledge we have about the world).

6.1.4 From sets to space of models

So far we've talked about **sets** of *models*, that on our Bayesian view are fluctuating around the data. But when we talk about *comparing models*, we necessarily have to recognize some sort of structure in the space where all the models live. By introducing a **topology** we can't already put a number on each model, but we start to have the notion of closeness between two different models, namely how similar they are. The next step is to put a **measure** in the models' space, so that one can say precisely how far two models are from each other.

■ **Example 6.1 — Gull's Decay Length Problem.** The following problem is taken from [Mac02], chapter 3.

Unstable particles are emitted from a source and decay at distance x , a real number that has an exponential probability distribution with a characteristic length, λ :

$$p(x|\lambda) = \frac{e^{-\frac{x}{\lambda}}}{Z} \quad (6.1)$$

Decay events can only be observed if they occur in the window extending from $x = a = 1$ cm to $x = b = 20$ cm. N decays are observed at locations $\{x_1, \dots, x_N\}$. What is λ ? We're going to go through the solution later

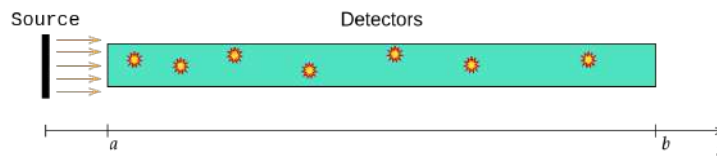


Figure 6.4: Configuration example

on. For the moment, let us focus on what we have here. The likelihood we have above depends on just one parameter, λ , and it's quite remarkable that we can describe the whole phenomenon with just one number! From the perspective we're building, we can think of λ as an index for all possible models, rather than just a

parameter to fit on the data we collect. Also, since λ is a positive real number, we automatically inherit the topology and the notion of distance that we have in \mathbb{R} , allowing us to compare easily different models. Of course, it's not always that easy. When we want to compare more complicated parametric models with more than one parameter, we will see that a good distance is given by the Fisher metrics. ■

The "Ball in a Box" analogy

Another way to visualize this Bayesian reasoning is depicted in Figure 6.5. The cloud on the right represents all the data we can potentially observe from the experiment, for example all the measurements we can make in the interval $x \in [1, 20]$ cm in the Gull's problem presented above. The blue ball around the set of data we took represents the *uncertainty* that we have on our measurements. It can be devoted for example to the non-infinite accuracy of our detectors, and it can actually change shape within the different regions of the data space. Then, the Bayesian inference takes the observations and "pulls them back" into the model space, the interval $(0, +\infty)$ where λ lives. This procedure produces a lot of points, which eventually create the green ellipse concentrate around the best estimate we can give for the parameter.

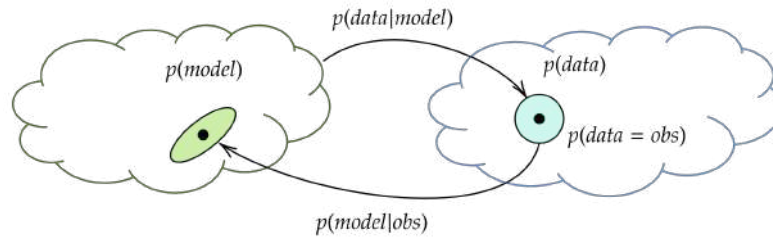


Figure 6.5: Mapping between two spaces

6.1.5 Parameters estimation: Gull's problem solution

Let's go through the resolution of the Gull's decay length problem. The Bayes' formula we want to apply is nothing but the green lips into the space of models that we have on figure 6.5. The blue ball is instead the likelihood (6.1), and it tells us how a particular choice of λ looks like over the data space

$$p(x|\lambda) = \frac{e^{-\frac{x}{\lambda}}}{Z}, \quad Z(\lambda; a, b) = \int_a^b dx e^{-\frac{x}{\lambda}} = \lambda \left(e^{-\frac{a}{\lambda}} - e^{-\frac{b}{\lambda}} \right)$$

$$\xrightarrow{N \text{ samples}} p(\{x\}|\lambda) = \prod_{n=1}^N p(x|\lambda) = \frac{e^{-\frac{\langle x \rangle}{\lambda}}}{\lambda^N \left(e^{-\frac{a}{\lambda}} - e^{-\frac{b}{\lambda}} \right)^N} \quad \text{Likelihood}$$

where it is worth to notice that the this function is normalized over x . Now we have to choose a prior. Since we know that our measurements lie in the interval $[a, b]$, a simple choice is to use a uniform (flat) prior. Another possibility is given by the gamma distribution, but in this case is not trivial how to set the parameters of the distribution. The ideal scenario it that we have so many measurements that the particular choice of the prior doesn't matter anymore.

$$p(\lambda|a, b) = \frac{1}{b-a} \mathbb{1}_{[a, b]} \quad \text{Uniform prior}$$

$$p(\lambda|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda} \quad \text{Gamma prior}$$

Let's consider the uniform prior. The **evidence** is very often computed via numerical integration (or Monte Carlo methods), but here we can write it in the analytical form:

$$p(\{x\}) = \int_0^\infty d\lambda p(\{x\}|\lambda) p(\lambda) = \frac{1}{b-a} \int_a^b d\lambda \frac{e^{-\frac{\langle x \rangle}{\lambda}}}{\lambda^N \left(e^{-\frac{a}{\lambda}} - e^{-\frac{b}{\lambda}} \right)^N} \leftarrow \left(\begin{array}{l} \lambda = u^{-1} \\ d\lambda = -u^{-2} du \end{array} \right)$$

$$= \frac{1}{b-a} \int_{a^{-1}}^{b^{-1}} du u^{N-2} \frac{e^{\langle x \rangle u}}{(e^{-au} - e^{-bu})^N} \quad \text{Evidence}$$

Putting all pieces together, we get the Posterior

$$p(\lambda|\{x\}) = \frac{p(\{x\}|\lambda)p(\lambda)}{p(\{x\})} = \frac{e^{-\frac{\langle x \rangle}{\lambda}}}{\lambda^N \left(e^{-\frac{a}{\lambda}} - e^{-\frac{b}{\lambda}} \right)^N \int_{a^{-1}}^{b^{-1}} du u^{N-2} \frac{e^{\langle x \rangle u}}{(e^{-au} - e^{-bu})^N}} \mathbb{1}_{[a,b]} \quad \text{Posterior}$$

6.1.6 Parameters estimation: Bretthorst' spectral analysis

In this section we report few steps taken from the article [Bre08], just to emphasize how Bayesian inference can be exploited to do parameters estimation. We recommend to have a look at the source for a complete understanding of the problem.

The problem of estimating the frequency of a sinusoid occurs in many different areas of science and engineering. From the standpoint of estimating a frequency, uniform samples are not more informative than nonuniform samples; they are more convenient because of the almost universal use of the fast discrete Fourier transform in the frequency estimation procedure. We will see how probability theory generalizes the discrete Fourier transform.

Suppose that we are taking some astronomical data, and our apparatus give us the complex and the real part of the signal; by the fact that data points are taken at different times, then we have two different data sets, $D_R \equiv \{d_R(t_1) \dots d_R(t_{N_R})\}$ and $D_I \equiv \{d_I(t_1) \dots d_I(t_{N_I})\}$. The actual measurement data is therefore

$$\tilde{d}(t_j) = d_R(t_j) + i d_I(t_j)$$

We impose no restrictions on the number of data samples or the acquisition times in either channels. If the complex data are given by the equation above, then the data and the sinusoid are related by

$$\tilde{d}(t_j) = \tilde{A} \exp\{-\tilde{f}(t_j)\} + \tilde{n}(t_j)$$

where the complex amplitude is given by $\tilde{A} = A_1 - iA_2$, and is equivalent to the amplitude and phase of the sinusoid. The complex frequency, $\tilde{f} = \alpha + 2\pi i f$, contains two parameters: the decay rate constant, α , and the frequency, f . The quantity $\tilde{n}(t_j)$ represents the complex noise at time t_j . Note that in this equation the times t_j , simply designate the times at which we actually have data. Consequently we can split our data sample $d_R(t_i)$ and $d_I(t_j)$ as

$$d_R(t_i) = M_R(t_i) + n_R(t_i) \quad d_I(t_j) = M_I(t_j) + n_I(t_j)$$

where

$$M_R(t_i) = \left[A_1 \cos(2\pi f t_i) - A_2 \sin(2\pi f t_i) \right] \exp\{-\alpha t_i\}$$

$$M_I(t_j) = - \left[A_1 \sin(2\pi f t_j) + A_2 \cos(2\pi f t_j) \right] \exp\{-\alpha t_j\}$$

In probability theory as logic, all of the information about an hypothesis is summarized in a probability density function, in our case $P(f, \alpha | D_R, D_I, I)$. We can obtain such pdf marginalizing over parameters that we are really not interested about:

$$P(f, \alpha | D_R, D_I, I) = \int dA_1 dA_2 d\sigma P(D_R, D_I | f, \alpha, A_1, A_2, \sigma, I) P(f, \alpha, A_1, A_2, \sigma | I)$$

If the two data sets are logically independent, the joint direct probability for the data will factorize as

$$P(D_R, D_I | f, \alpha, A_1, A_2, \sigma, I) = P(D_R | f, \alpha, A_1, A_2, \sigma, I) \times P(D_I | f, \alpha, A_1, A_2, \sigma, I)$$

We will assume that the prior can be factorized into independent prior distributions for each parameter. These will be uniform priors for the amplitudes, frequency and decaying rate, and a Jeffreys' prior for the standard deviation of the noise.

If for the likelihood we use Gaussians, we have

$$P(f, \alpha | D_R, D_I, I) \propto \int dA_1 dA_2 \frac{d\sigma}{\sigma} \times \sigma^{-N_R} \exp \left\{ -\frac{1}{2\sigma^2} \sum_{i=0}^{N_R-1} [d_R(t_i) - M_R(t_i)]^2 \right\} \\ \times \sigma^{-N_I} \exp \left\{ -\frac{1}{2\sigma^2} \sum_{j=0}^{N_I-1} [d_I(t_j) - M_R(t_j)]^2 \right\}$$

Finally we obtain

$$P(f, \alpha | D_R, D_I, I) \propto \int dA_1 dA_2 d\sigma \sigma^{-(N_R+N_I+1)} \exp \left\{ -\frac{Q}{2\sigma^2} \right\} \propto \frac{1}{\sqrt{ab-c^2}} \left[(N_R+N_I)\overline{d^2} - \overline{h^2} \right]^{1-\frac{N_R+N_I}{2}} \quad (6.2)$$

where $Q, a, b, c, \overline{h^2}, \overline{d^2}$ are particular summary statistic of the data, with

$$\overline{d^2} = \frac{1}{N_R+N_I} \left[\sum_{i=0}^{N_R-1} d_R(t_i)^2 + \sum_{j=0}^{N_I-1} d_I(t_j)^2 \right]$$

The interesting thing is that $\overline{h^2}$ is a summary statistic that generalizes the discrete Fourier transform power spectrum to nonuniformly nonsimultaneously sampled data.

If one is interested only in estimating the frequency or the decaying rate, he can always marginalize over the other parameters; moreover if we know that absolutely there is no decaying in the signal we can just substitute $\alpha = 0$ and $N_I = 0$, meaning no imaginary part of the signal. Our calculation then reduce to:

$$P(f|D, I) \propto \frac{1}{\sqrt{ab-c^2}} \left[N\overline{d^2} - \overline{h^2} \right]^{1-\frac{N}{2}} \quad \text{where } \overline{d^2} = \frac{1}{N} \sum_{n=1}^N d_n^2 \quad (6.3)$$

6.2 Conjugate priors

When facing Bayesian inference problems, we usually get trapped into really nasty integrals. We also had better avoid Monte Carlo methods to solve them: they must be seen as a parachute, used only in case of emergency. One way to choose the path of **mathematical convenience** is to find the right matching between the likelihood and the prior, selecting our distribution for the parameters from the family of **conjugate priors**. In the following, we're going to present two examples: the first one represents the family of discrete distributions, whereas the second one is taken from the continuous distributions' family.

6.2.1 Discrete distributions: solving the Coin Tossing problem

Considering the toss of a coin, we want to evaluate whether the coin is fair or not. Our goal is therefore to infer the probability B (bias) to get a head (H) from a toss and to verify the compatibility with the fair value $B = 1/2$. The likelihood for a single toss is given by $p(x = H|B) = B$, and consequently $p(x = T|B) = 1 - B$. Our data consist in the number of heads N_H obtained in N trials, and the likelihood is given by the binomial distribution:

$$p(N_H|B, N) = \binom{N}{N_H} B^{N_H} (1-B)^{N-N_H} \quad \text{Likelihood}$$

In order to exploit the mathematical properties of the conjugate priors, we choose the beta distribution

$$p(B|\alpha, \beta) = \frac{B^{\alpha-1} (1-B)^{\beta-1}}{\mathcal{B}(\alpha, \beta)}, \quad \text{Prior}$$

in which $\mathcal{B}(\alpha, \beta)$ is the beta function

$$\mathcal{B}(\alpha, \beta) = \int_0^1 x^{\alpha-1} (1-x)^{\beta-1} dx$$

The evidence can be computed directly by integration,

$$p(N_H|N; I) = \int_0^1 p(N_H|B, N) p(B|\alpha, \beta) dB = \binom{N}{N_H} \frac{\overbrace{\int_0^1 dB B^{N_H+\alpha-1} (1-B)^{N-N_H+\beta-1}}^{\mathcal{B}(N_H+\alpha, N-N_H+\beta)}}{\mathcal{B}(\alpha, \beta)} \quad \text{Evidence}$$

and the Bayes' theorem gives us the posterior

$$p(B|N_H, N; I) = \frac{p(N_H|B, N)p(B|\alpha, \beta)}{p(N_H|N; I)} = \frac{B^{N_H+\alpha-1}(1-B)^{N-N_H+\beta-1}}{\mathcal{B}(N_H+\alpha, N-N_H+\beta)} \quad \text{Posterior}$$

In the above expression, we observe that we can get the posterior starting from the prior simply through an **hyper-parameters update**

$$\begin{aligned}\alpha &\rightarrow \alpha' \equiv \alpha + N_H \\ \beta &\rightarrow \beta' \equiv \beta + N - N_H = \beta + N_T\end{aligned}$$

In this way, we can interpret the inference as a dynamical process in the manifold of beta distributions, where the dynamics is pushing our prior parameters toward two different α' , β' in the manifold (figure 6.6). The mathematical convenience of choosing a prior within the conjugate family of a given likelihood is clear: all the inference problem reduces to the update of few numbers, and we can spare ourselves the effort of computing nasty integrals. Moreover, the Beta distribution has a very versatile behaviour when changing the parameters α and β , ranging from a **uniform distribution** ($\alpha = \beta = 1$) to very peaked shapes (see Figure 6.7).

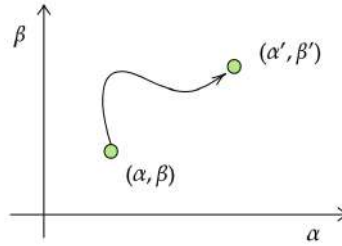


Figure 6.6: The inference process can be seen as a trajectory in the models' space

At this point we can give an estimate of the most probable value of B through the **Maximum A Posteriori (MAP) Estimator**, i.e. by evaluating the value of maximum of the posterior; we define $a = N_H + \alpha - 1$ and $b = N - N_H + \beta - 1$, so

$$\begin{aligned}0 &= \frac{d}{dB} B^a (1-B)^b = aB^{a-1}(1-B)^b - bB^a(1-B)^{b-1} \\ &= (aB^{-1} - b(1-B)^{-1})B^a(1-B)^b = 0 \\ &\iff a\hat{B}^{-1} - b(1-\hat{B})^{-1} \\ &\implies a(1-\hat{B}) = b\hat{B} \\ &\implies a = (a+b)\hat{B} \\ &\implies \hat{B} = \frac{a}{a+b}\end{aligned}$$

In our case:

$$\hat{B}_{MAP} = \frac{N_H + \alpha - 1}{N + \alpha + \beta - 2} \quad \text{MAP}$$

The corresponding frequentist estimator is the **Maximum Likelihood Estimator (MLE)**, that in our case returns

$$\hat{B}_{MLE} = \frac{N_H}{N} \quad \text{MLE}$$

Notice that the two estimators coincide if we choose a uniform prior (i.e. selecting $\alpha = \beta = 1$). From this fact we can conclude that the MLE method hides some implicit mathematical prescription for our ignorance, whereas the Bayesian approach makes the choice of the prior explicit!

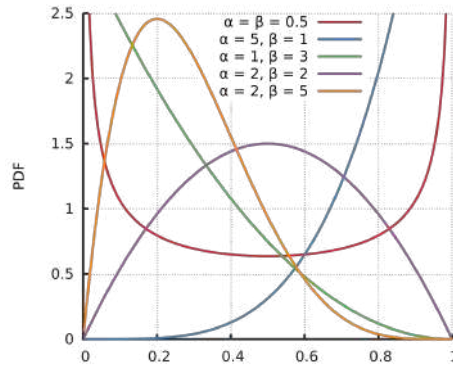


Figure 6.7: Beta distribution for different values of parameters α and β . Source: https://en.wikipedia.org/wiki/Beta_distribution

6.2.2 Continuous distributions: Gaussian-shaped likelihood

Consider a Gaussian likelihood evaluated over n observations $\{x_i\}_{i=1\dots n} \equiv \mathcal{D}$

$$\mathcal{N}(\mathcal{D}|\mu, \sigma^2) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x_i - \mu)^2}{2\sigma^2}\right] \quad \text{Likelihood}$$

The conjugate priors for the parameters of such a likelihood, turn out to be an inverse gamma distribution over σ^2 (or a gamma distribution over $\lambda = 1/\sigma^2$) and a Gaussian PDF over μ

$$p(\mu|\sigma^2; \mu_0, \kappa_0) = \sqrt{\frac{\kappa_0}{2\pi\sigma^2}} \exp\left[-\frac{\kappa_0}{2\sigma^2}(\mu - \mu_0)^2\right] \quad \mu \text{ Prior}$$

$$p(\sigma^2|\alpha = \nu_0, \beta = \sigma_0^2) \propto \frac{\sigma_0^2}{\Gamma(\nu_0)} \left(\frac{\sigma_0^2}{\sigma^2}\right)^{\nu_0-1} \exp\left(-\frac{\sigma_0^2}{\sigma^2}\right) \quad \sigma^2 \text{ Prior}$$

The total normalized prior is therefore given by

$$p(\mu, \sigma^2|\mu_0, \kappa_0, \nu_0, \sigma_0^2) = \sqrt{\frac{\kappa_0}{2\pi\sigma^2}} \frac{1}{\sigma_0^2 \Gamma(\nu_0)} \left(\frac{\sigma_0^2}{\sigma^2}\right)^{\nu_0+1} \exp\left[-\frac{2\sigma_0^2 + \kappa_0(\mu - \mu_0)^2}{2\sigma^2}\right]$$

which is also called *Normal-Inverse-Gamma (NIG) distribution*. The Bayes' theorem reads

$$p(\mu, \sigma^2|\mu_n, \kappa_n, \nu_n, \sigma_n^2) = \frac{1}{Z} \mathcal{N}(\mathcal{D}|\mu, \sigma^2) p(\mu, \sigma^2|\mu_0, \kappa_0, \nu_0, \sigma_0^2) \quad \text{Posterior}$$

The proof of the parameters' update rule goes through the computation of the normalization factor Z , and here we omit the details. In the end one gets

$$\begin{aligned} \kappa_n &= \kappa_0 + n & \nu_n &= \nu_0 + \frac{n}{2}; \\ \mu_n &= \frac{\kappa_0\mu_0 + n\bar{x}}{\kappa_0 + n}; & \sigma_n^2 &= \sigma_0^2 + \frac{ns^2}{2} + \frac{1}{2} \frac{n\kappa_0}{\kappa_0 + n} (\bar{x} - \mu_0)^2 \end{aligned}$$

where

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i; \quad s^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

Once we define the *precision* $\lambda = 1/\sigma^2$, we can visualize the iso-contours of the probability distribution in the $\mu - \lambda$ space, as shown in Figure 6.8. This plot allows us to see what is our ignorance about our parameters. As we add some data, hopefully the contour plot will get narrower around some point in the parameters' space.

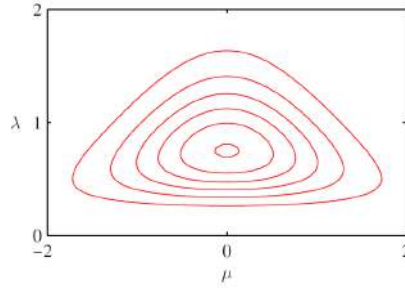


Figure 6.8: Contour plot in the parameters' space that quantify our ignorance about the parameters of the model. Image taken from [Bis06], p.102

6.2.3 Predictive posteriors

Another field we can get into is the one of **predictive posteriors**: once we have computed the posterior for the parameters of the model, we try to forecast the next observation \tilde{x} given all the information we collected until that point, namely our sample $\mathbf{X} = (x_1, \dots, x_n)$. In other words, the posterior predictive distribution is the distribution of possible unobserved values conditioned on the observed values:

$$p(\tilde{x}, \mathbf{X}) = \int_{\Theta} p(x, \boldsymbol{\theta} | \mathbf{X}) d\boldsymbol{\theta} = \int_{\Theta} p(x | \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathbf{X}) d\boldsymbol{\theta}$$

In the specific case we're dealing with, if we take a Gaussian $\mathcal{N}(x | \mu, \sigma^2)$, we multiply it by the posterior found above and we integrate out μ and σ^2 , in the end what we get is

$$\begin{aligned} p(\tilde{x} | \mu_n, \kappa_n, \nu_n, \sigma_n^2) &= \int_0^\infty d\sigma^2 \int_{-\infty}^\infty d\mu \mathcal{N}(\tilde{x} | \mu, \sigma^2) p(\mu, \sigma^2 | \mu_n, \kappa_n, \nu_n, \sigma_n^2) \\ &= \frac{1}{\sqrt{2\pi\sigma_n^2}} \frac{\Gamma(\nu_n + 1/2)}{\Gamma(\nu_n)} \sqrt{\frac{\kappa_n}{\kappa_n + 1}} \left(1 + \frac{\kappa_n}{\kappa_n + 1} \frac{(\tilde{x} - \mu_n)^2}{2\sigma_n^2} \right)^{-(\nu_n + \frac{1}{2})} \end{aligned} \quad (6.4)$$

The Student-t distribution is defined as

$$t_\nu(\tilde{x} | \mu, \sigma^2) = \frac{\Gamma(\nu/2 + 1/2)}{\Gamma(\nu/2)} \frac{1}{\sqrt{\nu\pi\sigma}} \left[1 + \frac{1}{\nu} \left(\frac{x - \mu}{\sigma} \right)^2 \right]^{-\frac{\nu+1}{2}}$$

where the parameter ν is called *degrees of freedom*, and it's known that the Student-t distribution tends to a Gaussian in the limit $\nu \rightarrow \infty$, as shown in figure 6.9.

If we now look at Eq. (6.4), we immediately realize that **the predictive posterior for a Gaussian random process with mean and variance unknown is given by the Student-t distribution**

$$p(\tilde{x} | \mu_n, \kappa_n, \nu_n, \sigma_n^2) = t_{2\nu_n} \left(\tilde{x} | \mu_n, \frac{\sigma_n^2(\kappa_n + 1)}{\kappa_n \nu_n} \right)$$

The Student-t distribution is a smartest version of the Gaussian for describing our data. Indeed, the thin tails of the Gaussian are very sensible to outliers, and when we have few data the distribution strives to adapt his shape to accomodate them. Instead, the Student-t has very fat tails when the number of samples is small, meaning that the appearance of outliers is not a big deal.

This is a sort of Bayesian interpretation of why we do the Student-t testing for finite number of samples, and it's definitely more satisfying than all the tricky reasoning one has to do in the frequentist approach.

For a nice summary of all the discrete and continuous conjugate priors and predictive posteriors we recommend the Wikipedia's page https://en.wikipedia.org/wiki/Conjugate_prior.

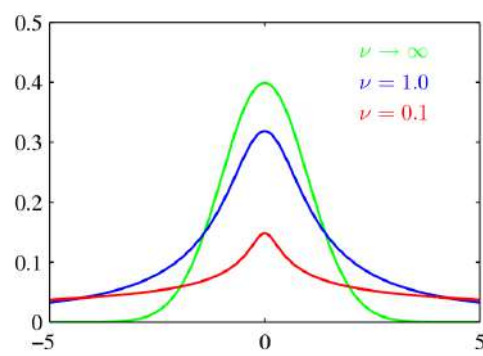
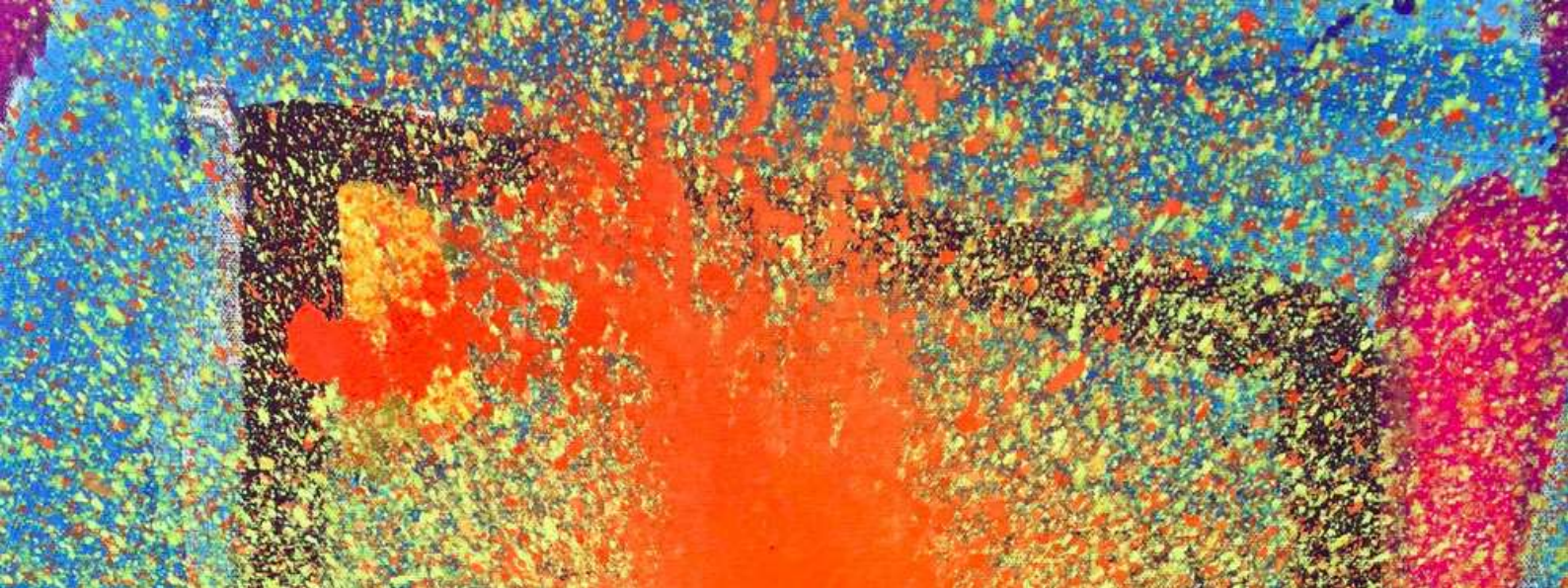


Figure 6.9: The Student-t distribution for different values of ν . In the limit $\nu \rightarrow \infty$ we retrieve a Gaussian distribution. Figure taken from [Bis06], p. 103



7. Entropy and Information

7.1 Learning by diffusing: the information potential

Lesson 12

30/04

GC

FC

Let's start by considering a physical problem, where a bunch of particles are moving within an unknown potential, $u(\mathbf{x})$. In general, the aim of physics is to study population dynamics given the potential; however, in the inference problem things go in the opposite direction: the shape of $u(\mathbf{x})$ is what we want to infer starting from the position of the particles.

One possible way to solve this problem is to place the N particles in some chosen points in the space and study their evolution. Starting from the initial state, by using the drift-diffusion equation we can get the final equilibrium state.

$$\text{IC: } p(\mathbf{x}, 0) = \frac{1}{N} \sum_{n=1}^N \delta(\mathbf{x} - \mathbf{x}_n) \quad \partial_t p = \nabla \cdot (D \nabla p + p \nabla u)$$

As we know from statistical mechanics, in the equilibrium state the probability distribution is Boltzmann-like:

$$\lim_{t \rightarrow \infty} p(\mathbf{x}, t) = Z^{-1} e^{-u(\mathbf{x})/D} \Rightarrow u(\mathbf{x}) = -D \ln p(\mathbf{x}) - D \ln Z$$

and we can see an interesting relation between the potential and the negative logarithm of the probability distribution. The constant term coming out from the partition function is actually not important, since we're always interested in difference of potential.

Starting from the assumption $u = 0$, the diffusion equation is solved by the Green function G , that defines the probability for a particle to be at position \mathbf{x} at time t given that we started at the point \mathbf{x}'

$$G(\mathbf{x}, \mathbf{x}'; t) = \frac{1}{(4\pi Dt)^{\frac{d}{2}}} \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{4Dt}\right)$$

To compute the probability distribution, we have first to determine the characteristic time t^* at which the flux of particles (and hence of probability) is zero: $\mathbf{J}(t^*) = 0$. Once we've done this, we can average the Green's function over the number of particles and get the estimated pdf

$$p(\mathbf{x}|t^*) = \frac{1}{N} \sum_{n=1}^N \frac{1}{(4\pi Dt^*)^{\frac{d}{2}}} \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_n\|^2}{4Dt^*}\right)$$

At this point it's possible to compute the final potential as $u(\mathbf{x}) = -D \ln p(\mathbf{x}|t^*)$. This physical approach allows us in some way to reverse our usual perspective, and see probability distributions as actual potentials in which particles are "trapped". The final result clearly depends on how many particles are involved in the simulation (see figure 7.1). This parameter can be seen as a **temperature**: in the limit of an infinite number of particles (infinite low temperature), the posterior will match the underlying undisclosed potential.

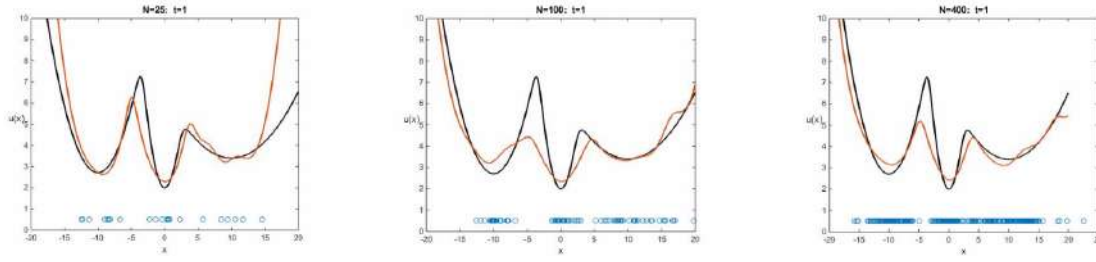


Figure 7.1: Estimation of the real potential (black curve) from the positions of the particles. From left to right, the number of samples has been increased, and the estimated curve (orange line) moves closer to the true potential. The resulting effect can be seen as a progressive lowering in temperature.

Figure 7.2 shows the beautiful interpretation of the process. Data have been generated according to a superposition of three Gaussian distributions, and particles are clustered around the regions of high probability. On the other hand, if we compute the information potential as the negative logarithm of the generating pdf, we can get a physical intuition of the process: the potential acts through a force on the particles that constrains them close to the potential's minima.

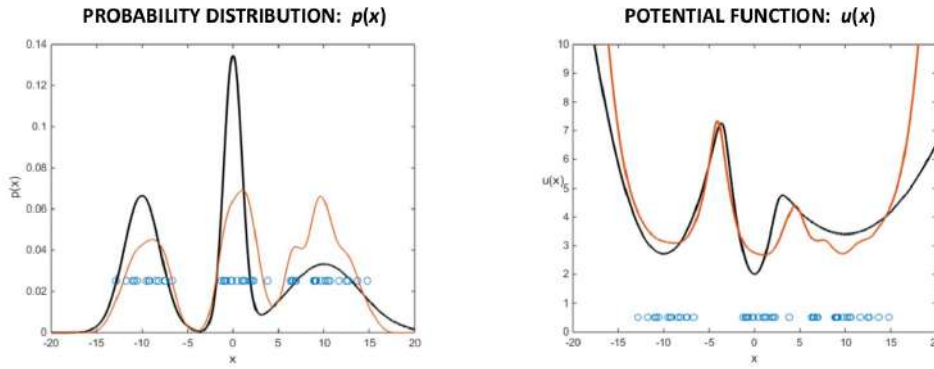


Figure 7.2: On the left, the black line represents the pdf used to generate the samples. On the right there is the corresponding information potential

7.2 Distance measure in information theory

In order to understand how different is a probability distribution from another one, we need to define some functions that can actually measure “distances” between them. There are many different ways we can use to carry out this task and we report here some of them:

- Root mean square error (RMSE):

$$D_{L2}(p, q) = \left(\int dx |p(x) - q(x)|^2 \right)^{\frac{1}{2}}$$

- Hellinger distance:

$$D^0(p, q) = 2 \int dx \left(\sqrt{p(x)} - \sqrt{q(x)} \right)^2$$

This is quite interesting in quantum mechanics, where one would take the difference between the two wave functions and square it to get a probability distribution.

- Kullback-Leibler divergence (KL) (or Relative entropy):

$$D_{KL}(p||q) = \int dx p(x) (u_q(x) - u_p(x)) = \int dx p(x) \ln \frac{p(x)}{q(x)}$$

The Kullback-Leibler (KL) divergence is the most used one since it allows to express properly some information related properties of the probability distribution, as we will see later on.

We can now introduce some functions in order to deal with information. So, *given a sample, which information does it provide?* This can be quantified by the **self-information**:

$$i_p(\mathbf{x}) = \log \left(\frac{1}{p(\mathbf{x})} \right) = -\log p(\mathbf{x}) \quad (7.1)$$

Notice that this quantity resembles what we previously called information potential. A way to compare different distribution is by means of the **relative self-information**:

$$\delta_{p,q}(\mathbf{x}) = i_q(\mathbf{x}) - i_p(\mathbf{x}) = \log \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} \right) \quad (7.2)$$

and in our physical analogy it represents the difference in potential energy between two different distributions. If we average these two expressions over $p(\mathbf{x})$, we get respectively:

$$H = \langle i_p(\mathbf{x}) \rangle_p = - \int p(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x} \quad \text{Entropy}$$

$$D_{KL}(p||q) = \langle \delta_{p,q}(\mathbf{x}) \rangle_p = \int p(\mathbf{x}) (i_q(\mathbf{x}) - i_p(\mathbf{x})) d\mathbf{x} = \int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x} \quad \text{Relative Entropy}$$

The relative entropy compares two probability distributions, and in some way allows to know how far they are from each other. The relative entropy has the important property to be non-negative, and this can be proven using the Jensen's inequality. But **is not a distance**, since it doesn't satisfy some fundamental properties that a distance function must have, such as the symmetry property.

In the definitions listed above we used the base-2 logarithm, but often they're defined using the natural logarithm instead. In the first case they're measured in *bits*, and it is conceptually easier to interpret them as measures of information. The natural logarithm is used instead when we want to take derivatives, so that we avoid annoying prefactors arising in computations. In this case the unit of measure is called *nat*.

7.2.1 Self-information or Information Potential

What is the interpretation we can give to the expressions above? How can we get a feeling of the fact that Eq. (7.1) assigns higher content of information to the less probable events?

A simple example can be given by the uniform binning of a Gaussian (see figure 7.3). What we want to do is to distribute the events in such a way that each bin correspond to a total probability of $P_n = 2^{-L_n}$, with $\sum P_n \leq 1$ and L_n being an integer number. Using this assumption, it is possible to associate a bin to the corresponding probability and to binary code using an *Huffman Tree*¹.

The idea is that each bifurcation of the tree corresponds to a decision to take; each of the two possible paths is labelled with "1" or "0", and they have the same probability to be chosen. As we proceed downward the tree, we collect the bits corresponding to the decisions we take, and at the bottom we end up in one of the possible bins of the Gaussian. Since each branch is equally probable, it's clear that the more bifurcations we encounter, the more decisions we have to take and the less probable the bin we fall in will be. At the same time, to get an unlikely bin we have also to collect many bits, and so the amount of information associated is also large.

There is another interesting observation to point out. If we plot the number of bits L_n (i.e. the number of decisions to take) as a function of the position in the Gaussian, x , what we find is a parabolic shape (figure 7.4, on the left). But L_n , which is measured in bits, is also the negative logarithm of the probability, and therefore it is nothing but the information potential! This is again showing us how the information content of the data is encoded in this "minus the logarithm of the probability".

7.2.2 Decomposition of the Entropy

Let us illustrate an interesting property of the entropy through an example. Imagine that a random variable $x \in \{0, 1, 2\}$ is created by first flipping a fair coin to determine whether $x = 0$; then, if x is not 0, flipping a

¹A nice video for understanding Huffman codes is <https://youtu.be/B3y0RsVCyrw>

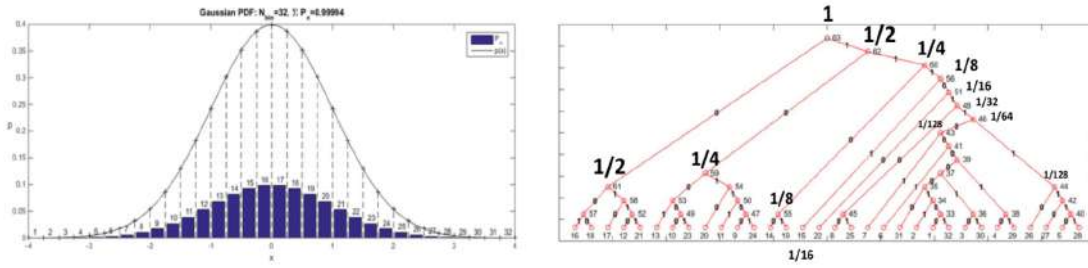


Figure 7.3: Huffman tree example

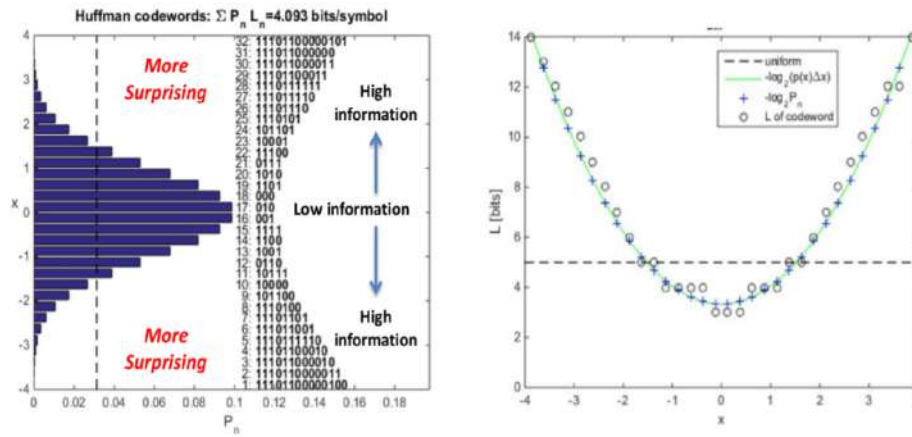


Figure 7.4: Left) Each bin of the Gaussian is associated with an information content. The more unlikely is the bin, the longer the sequence of bit to describe it will be. Right) The negative logarithm of the probability gives the information potential, that in this case is a parabola.

fair coin a second time to determine whether x is 1 or 2. The probability distribution of x is

$$p(X=0) = \frac{1}{2} \quad p(X=1) = \frac{1}{4} \quad p(X=2) = \frac{1}{4}$$

What is the entropy of X ? We can either compute it by brute force

$$H(X) = \frac{1}{2} \log 2 + \frac{1}{4} \log 4 + \frac{1}{4} \log 4$$

or we can use the following decomposition, in which the value of x is revealed gradually.

Imagine first to learn whether $x = 0$ or not, and then, if x is different from 0, to learn which non-zero value is the final outcome. The discovery of whether $x = 0$ or not entails revealing a binary variable whose probability distribution is $\{1/2, 1/2\}$. This revelation has an entropy $H(1/2, 1/2) = \frac{1}{2} \log 2 + \frac{1}{2} \log 2 = 1$ bit. If x is not 0, we learn the value of the second coin flip. This is again a binary variable whose probability distribution is $\{1/2, 1/2\}$, and whose entropy is 1 bit. We only get to experience the second revelation half the time, however, so the entropy can be written:

$$H(X) = H\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{1}{2} H\left(\frac{1}{2}, \frac{1}{2}\right)$$

Generalizing, the observation we are making about the entropy of any probability distribution $p = \{p_1, p_2, \dots, p_I\}$ is that

$$H(\mathbf{p}) = H(p_1, 1-p_1) + (1-p_1) H\left(\frac{p_2}{1-p_1}, \frac{p_3}{1-p_1}, \dots, \frac{p_I}{1-p_1}\right)$$

7.2.3 Joint entropy and conditional entropy

In what follows we define some quantities which we'll have to deal with and we prove some of their properties.

Definition 7.1 — Joint entropy. The *joint entropy* $H(X, Y)$ of a pair of discrete random variables (X, Y) with a joint distribution $p(x, y)$ is defined as

$$H(X, Y) = - \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(x, y) \quad (7.3)$$

We also define the conditional entropy of a random variable given another as the expected value of the entropies of the conditional distributions, averaged over the conditioning random variable.

Definition 7.2 — Conditional entropy. If $(x, y) \sim p(x, y)$, the *conditional entropy* $H(Y|X)$ is defined as

$$\begin{aligned} H(Y|X) &= \sum_{x \in \mathcal{X}} p(x) H(Y|X = x) \\ &= - \sum_{x \in \mathcal{X}} p(x) \sum_{y \in \mathcal{Y}} p(y|x) \log p(y|x) \\ &= - \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(y|x) \end{aligned} \quad (7.4)$$

This measures the average uncertainty that remains about y when x is known.

The naturalness of the definition of joint entropy and conditional entropy is exhibited by the fact that the entropy of a pair of random variables is the entropy of one plus the conditional entropy of the other. This is proved in the following

Proposition 7.1 — Chain rule. $H(X, Y) = H(X) + H(Y|X) = H(Y) + H(X|Y)$

Proof.

$$\begin{aligned} H(X, Y) &= - \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(x, y) \\ &= - \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log [p(y|x)p(x)] \\ &= - \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(x) - \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(y|x) \\ &= - \sum_{x \in \mathcal{X}} p(x) \log p(x) - \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(y|x) \\ &= H(X) + H(Y|X) \end{aligned}$$

More directly, we can also write

$$\log p(X, Y) = \log p(X) + \log p(Y|X)$$

and take the expectation of both sides to obtain the claim of the proposition. The other version follows from the symmetry of the joint entropy. ■

7.2.4 Kullback-Leibler Divergence or Relative Entropy

As we said the KL allows us to measure distances, and so it can be used to check whether data generated from a model are able to reproduce the data we observed in Nature. We can even go further with this idea and minimize the KL **in order to find the best pdf for the given data**.

Here we want to study closer what is the behaviour of this quantity in different situations

$$D_{KL}(p||q) = \int p(\mathbf{x}) \ln \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x}$$

Usually we use to think that $p(\mathbf{x})$ is the *probability of the data*, which in general is not given, whereas $q(\mathbf{x})$ is the probability distribution of what the model thinks the data belong to. However, what we can do is to

generate synthetic data according to some known distribution, and to compare it with the distribution inferred by the model through KL-minimization. We choose $p(\mathbf{x})$ (green) as a multivariate Gaussian with a non-trivial covariance matrix $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mu, \Sigma)$, and $q(\mathbf{x})$ (red) as a simpler symmetric Gaussian $q(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mu, \sigma^2 \mathbb{1}_2)$. In figure 7.5 we can see what are the effects in minimizing $D_{KL}(p||q)$ rather than $D_{KL}(q||p)$. What we see is that $D_{KL}(p||q)$ tends to fit tighter the high probability regions of the distributions, whereas the minimization of $D_{KL}(q||p)$ gives more importance to the external regions, where the probability is lower. This very different behaviour shows why the KL divergence cannot actually be seen as a distance: it is not symmetric.

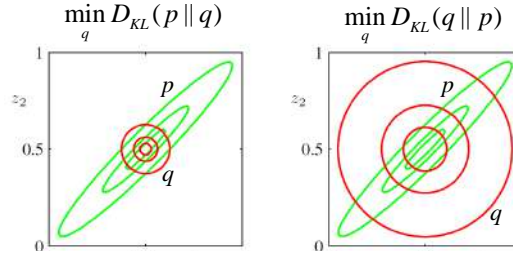


Figure 7.5: Inference of $p(\mathbf{x})$ starting from different definitions of the KL divergence. Image from [Bis06], p. 468

In figure 7.6 we repeat this experiment, this time with data sampled from a multimodal-multivariate normal distribution. In this case

Data: $p(\mathbf{x}) = \pi_1 \mathcal{N}(\mathbf{x}|\mu_1, \Sigma_1) + \pi_2 \mathcal{N}(\mathbf{x}|\mu_2, \Sigma_2)$

Model: $q(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mu_1, \Sigma_1)$

We see that $D_{KL}(p||q)$ tries to cover as much as possible the region where the two modes lie, whereas the minimization of $D_{KL}(q||p)$ can get stuck in a local minimum. This shouldn't be seen in general as a disadvantage, though; the choice of the KL depends on what one wants to do.

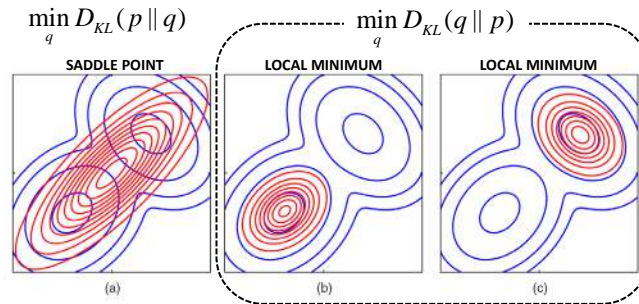


Figure 7.6: The two kinds of KL divergence have very different behaviours with multimodal distributions. Source: [Bis06], p. 469

7.2.5 Alternative measure functions

The KL is without any doubt the main function to express a distance between probabilities, but we may use different and more general kind of measures such as f -divergences or α -connections. In particular we can consider the class of α -divergences defined as

$$D^\alpha(p||q) = \frac{4}{1-\alpha^2} \left(1 - \int dx p(x)^{\frac{1+\alpha}{2}} q(x)^{\frac{1-\alpha}{2}} \right)$$

A different α gives different measure functions, for example:

- Hellinger distance squared: $D^0(p||q)$
- Dual KL: $D^{-1}(p||q)$
- KL: $D^1(p||q)$

In particular by using high α s the weight of $p(x)$ is increased since its exponent has a positive sign for α , while the situation is the opposite with very low or negative value of α .

7.2.6 Mutual information

In the Bayesian context it's often useful to measure **how much information is shared between two random variables X and Y** . In other words, we could ask: *how much information will we get on x if we observe y ?* A possible way to estimate this quantity is to use the mutual information, that allows us to understand how far away is a joint probability distribution of two *pdfs* from the product of the two marginalized distributions

$$I(X, Y) = D_{KL}(p(x, y) || p(x)p(y)) = \sum_{y \in \mathcal{A}_Y} \sum_{x \in \mathcal{A}_X} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \stackrel{(*)}{\geq} 0 \quad \text{Mutual information}$$

where $(*)$ is due to the Jensen's inequality, which in turn can be proved using the concept of convexity. This expression measures the non linear interdependence of variables x and y , and so how much information is shared between the two random processes. In the extreme case of two completely independent processes, the joint probability at the numerator of the ratio in the previous formula factorizes, which means that no mutual information is shared among the two ($\log 1 = 0$). In this case we cannot say anything about x by just measuring y .

Mutual information can be also re-expressed as a difference between entropies:

Proposition 7.2

$$\begin{aligned} I(X, Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X, Y) \end{aligned} \quad (7.5)$$

Proof.

$$\begin{aligned} I(X, Y) &= \sum_{x, y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \\ &= \sum_{x, y} p(x, y) \log \frac{p(x|y)}{p(x)} \\ &= - \sum_{x, y} p(x, y) \log p(x) + \sum_{x, y} p(x, y) \log p(x|y) \\ &= - \sum_x p(x) \log p(x) - \left(- \sum_{x, y} p(x, y) \log p(x|y) \right) \\ &= H(X) - H(X|Y) \end{aligned}$$

The second row follows from the symmetry, while to obtain the third row it is sufficient to apply the chain rule. ■

Since the entropy is the average information of a random process, from these expressions we can see that mutual information can be also interpreted as **the reduction in uncertainty over X after observing Y and vice versa**. The last row has instead a direct interpretation in the Eulero-Venn diagram shown in figure 7.7.

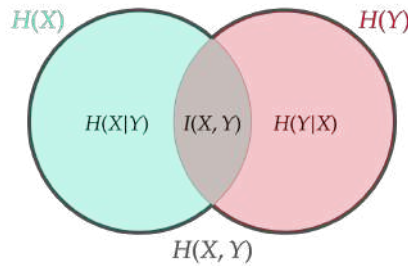


Figure 7.7: Eulero-Venn diagram for entropies and mutual information

From the proposition above we can obtain another important result

$$I(X, Y) \geq 0 \rightarrow H(X) \geq H(X|Y)$$

which means that observing the random variable Y cannot make the entropy of X increase, because we do learn something (or nothing) about it. This last result could be controversial, because sometimes it's possible to find

$$H(X) \leq H(X|Y = y).$$

However, this can happen only with single measurements, while on average learning X does convey information about Y .

Finally, we can also introduce the concept of conditional mutual information

Definition 7.3 — Conditional mutual information. The *conditional mutual information between X and Y given $z = c_k$* is the mutual information between X and Z in the joint ensemble $P(x, y|z = c_k)$

$$I(X|Y, z = c_k) = H(X|z = c_k) - H(X|Y, z = c_k)$$

By averaging over $z \in Z$ we obtain the *conditional mutual information between X and Y given Z*

$$I(X|Y, Z) = H(X|Z) - H(X|Y, Z)$$

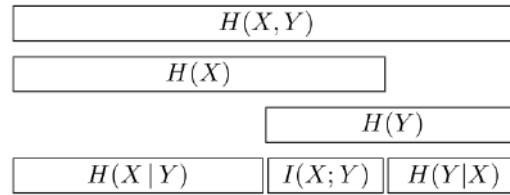


Figure 7.8: Graphical relationship between entropies, conditional entropies and mutual information of two random variables

S From the coding-computational point of view be aware that calculate quantities like $H(X, Y)$ could return NaN if the probabilities that we are computing are zero. A useful trick is to evaluate instead

$$H(X, Y) = - \sum_x \sum_y p(x, y) \log [p(x, y) + \varepsilon]$$

where ε could be the machine precision $\text{eps} = 2^{-56}$.

Covariance vs mutual information

It is worth to remark the difference between correlation and dependence between two random variables. It is well known that if X and Y are two independent random variables, then the covariance is zero $\langle XY \rangle = 0$. However, in real world problems things goes in the opposite direction. Bare in mind that *only* when our process is described by a Gaussian, i.e.

$$\mathcal{N}(X, Y) \propto \exp \left\{ -\frac{1}{2} (X, Y) \Sigma^{-1} (X, Y)^T \right\},$$

if the covariance is zero $\langle XY \rangle = 0$ then X and Y are actually independent. But this is exact only for the Gaussian pdf, whereas it is just an approximation if our distribution has a bell-like shape, so that we can express the real pdf as a Gaussian, plus some higher order corrections

$$f(X, Y) \propto \exp \left\{ -\frac{1}{2} (X, Y) \Sigma^{-1} (X, Y)^T + O(X^3 Y^3) \right\}.$$

This means that in general it's not true that if the covariance is zero then X and Y are independent: the covariance only keeps track of the simplest possible interdependence between X and Y , the linear one. The right tool to use in order to express our belief that X and Y have “something in common” is the mutual information $I(X, Y)$, because it retains all the possible moments of our distribution.

Information theoretic properties of functions of random variables

In some sense one could say that low complexity functions of random variables are resilient to noise, whereas high-complexity functions are easily degraded. There is a way of quantifying how much the “noise” destroys the deterministic mapping $f: x \rightarrow y$, and it is through the entropy. Indeed, if our function f is bijective, i.e. if every point x is mapped uniquely to some y and vice-versa, then $H(X) = H(f(X))$, i.e. the entropic content it's preserved.

Exercise — Entropy of functions of random variables

1. **Part 1.** Let X be a discrete random variable taking on a finite number of values. What is the relationship of $H(X)$ and $H(Y)$ if
 - (a) $Y = 2^X$
 - (b) $Y = \cos(X) \quad X \in [0, 2\pi)$
2. **Part 2.** Show that the entropy of a function of X is $H(X) \geq H(f(X))$ by following the steps below and justifying each one:
 - (a) $H(X, f(X)) = H(X) + H(f(X)|X) = H(X)$
 - (b) $H(X, f(X)) = H(f(X)) + H(X|f(X)) \geq H(f(X))$

Solution:

- 1a) The function is bijective, so if we know X we completely know also Y . This means that $H(X) = H(Y)$.
- 1b) Here the situation is different, because this is not an invertible function in $[0, 2\pi]$. Hence, it is still true that $H(Y|X) = 0$ because X still determines Y ; however $H(X|Y)$ is not zero, because for each value $y \in [-1, 1]$ there are two possible values of x . So knowing $Y = y$ tells you the set $\{x : x = \cos^{-1}(y)\}$, but it doesn't tell you what exactly X is.
- 2a) $H(X, f(X)) = H(X) + H(f(X)|X)$ for the chain rule (7.1), and $H(f(X)|X) = 0$ merely for the definition of function, so if we know X and f is deterministic then we know exactly $f(X)$.
- 2b) Again the chain rule (7.1) and then $H(X|f(X)) \geq 0$ for what we said before, where the equality holds if f is bijective.

7.3 Information theory version of Bayes' rule

Let us now reconsider the Bayes' formula. Suppose that we have a model depending on some parameters θ , and we also have some data y . The Bayes' rule tells us what is the probability distribution of parameters given the observations

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$

Now the question is: *how much have we learnt about the parameters of the model θ , having observed y ?* This is like asking what is the distance between our posterior and the prior, and we can quantify it through the KL divergence:

$$D_{KL}(p(\theta|y)||p(\theta)) = \sum_{\theta \in \Theta} p(\theta|y) \log \frac{p(\theta|y)}{p(\theta)} = \frac{1}{p(y)} \sum_{\theta \in \Theta} p(\theta, y) \log \frac{p(\theta, y)}{p(\theta)p(y)}$$

To make this result independent of the particular dataset we have, we shall average the previous expression over all possible realizations of the data we could measure from the process. This gives us the average information gained about the distribution of Θ by observing the distribution of Y , and it coincides with the mutual information between Θ and Y

$$I(\Theta, Y) = \sum_{y \in \mathcal{Y}} p(y) D_{KL}(p(\theta|y)||p(\theta))$$

Now we can use the property (7.5) to write the **information theory version of the Bayes' rule**

$$\boxed{H(\Theta|Y) = H(\Theta) - I(\Theta, Y)} \quad (7.6)$$

which is telling us that the learning process due to the inference acts by decreasing the entropy of the system by an amount equal to the mutual information between the two distributions.

Using this approach we can eventually choose the experimental design, which is represented by the analytical form of the likelihood, in such a way to **obtain the best information gain from inference**. This can be achieved by maximizing the mutual information between $\boldsymbol{\theta}$ and \mathbf{y}

$$\hat{p}(\mathbf{y}|\boldsymbol{\theta}) = \underset{p(\mathbf{y}|\boldsymbol{\theta})}{\operatorname{argmin}} \{H(\boldsymbol{\Theta}|\mathbf{Y})\} = \underset{p(\mathbf{y}|\boldsymbol{\theta})}{\operatorname{argmax}} \{I(\boldsymbol{\Theta}, \mathbf{Y})\}$$



8. Model Comparison

Suppose we collected a bunch of data and we want to describe them with a model. Most of the times it's not obvious what is the model that better describes what we observe, and we may try with different plausible proposals. What we need is therefore a method to decide which of those different models is the more appropriate for describing our observations.

Lesson 13
30/04
LR
TF

8.1 Occam's razor

Consider the drawing in figure 8.1. In foreground, there is a tree that doesn't allow to say whether what there's behind is one grey box or two separate smaller boxes. Is there a way to infer which of the two alternatives is the more probable? **Occam's razor** is a principle that states that given two hypothesis which are equally capable to describe the phenomenon, we should choose the simplest one. In this case we're prompted to choose the one-box hypothesis because, conscious or not, we realize that it would be a remarkable coincidence for the two boxes to be just the same height and colour of each other. In [Mac03], p. 351, there is a quantitative explanation of why this intuitive argument is actually essentially correct.

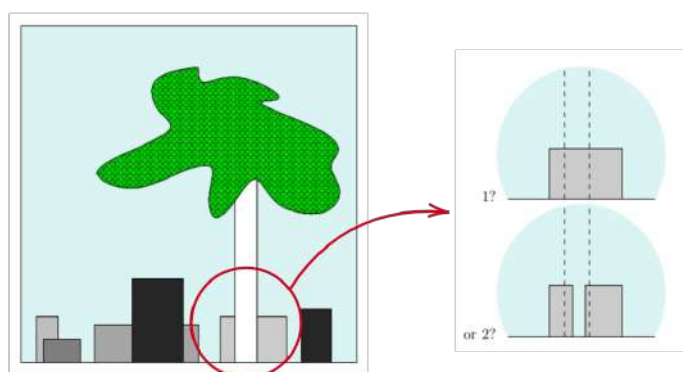


Figure 8.1: How many boxes are behind the tree?

8.1.1 Model comparison and Occam's razor

The Bayesian approach for model comparison automatically embodies Occam's razor, quantitatively. Suppose we want to evaluate the plausibility of two alternative theories, \mathcal{H}_1 and \mathcal{H}_2 in the light of data D . Using the Bayes' theorem, we relate the plausibility of the model \mathcal{H}_1 given the data, $P(\mathcal{H}_1|D)$, to the predictions made

by the model about the data, $P(D|\mathcal{H}_1)$, and the prior plausibility of \mathcal{H}_1 , $p(\mathcal{H}_1)$. This gives the following probability ratio between theory \mathcal{H}_1 and theory \mathcal{H}_2 :

$$\frac{P(\mathcal{H}_1|D)}{P(\mathcal{H}_2|D)} = \frac{P(\mathcal{H}_1)}{P(\mathcal{H}_2)} \cdot \frac{P(D|\mathcal{H}_1)}{P(D|\mathcal{H}_2)} \quad (8.1)$$

which represents how much \mathcal{H}_1 is favoured with respect to \mathcal{H}_2 . On the right side, the first ratio allows to insert a prior bias in favour of \mathcal{H}_1 , while the second ratio expresses how well the observed data were predicted by \mathcal{H}_1 , compared to \mathcal{H}_2 . The latter is also the term that embeds the Occam's razor: if \mathcal{H}_2 is a more complex model, it must spread its predictive probability $P(D|\mathcal{H}_2)$ more thinly over the data space than \mathcal{H}_1 (see figure 8.2). Thus, in the case where the data are compatible with both theories, the simpler \mathcal{H}_1 will turn out more probable than \mathcal{H}_2 .

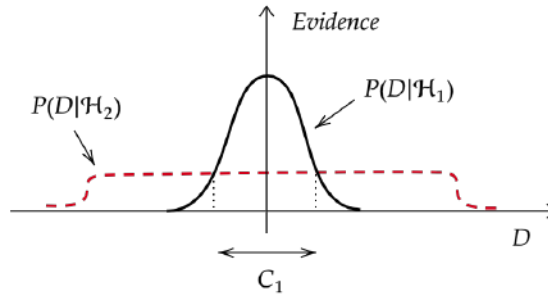


Figure 8.2: The predictions of the models are quantified by a normalized probability distribution on the space of possible data sets, D , which is called *Evidence*. Since each model has to spread his probability over D , we have that a simpler model, capable to make a limited range of predictions, is more probable than a more powerful one, as far as our data fall inside the region C_1 .

8.1.2 Evidence and the Occam's factor

In model comparison is not advisable to choose the model that fits the data best: more complex models can always fit the data better, so **the maximum likelihood model choice would led us inevitably to implausible, over-parametrized models, which generalize poorly.**

In Bayesian approach, we always go through **two levels of inference**. Suppose that each model \mathcal{H}_i is assumed to have a vector of parameters \mathbf{w} . Then, the two steps to follow are:

1. **Model fitting.** At first level of inference, we assume that one model, say \mathcal{H}_i , is true, and we infer what the model's parameters \mathbf{w} might be, given the data D . The *Posterior* probability of the parameters \mathbf{w} is

$$p(\mathbf{w}|D, \mathcal{H}_i) = \frac{P(D|\mathbf{w}, \mathcal{H}_i)P(\mathbf{w}|\mathcal{H}_i)}{P(D|\mathcal{H}_i)}. \quad (8.2)$$

The normalizing constant $P(D|\mathcal{H}_i)$ is commonly ignored in this first stage, because it is irrelevant to the end of inferring the posterior of \mathbf{w} ; however, it becomes important in the second stage of inference, and we name it **Evidence** for \mathcal{H}_i .

2. **Model comparison.** At the second level, we wish to infer which model is most plausible given the data. Applying the Bayes' theorem again, we can write the probability of each model as

$$P(\mathcal{H}_i|D) = \frac{P(D|\mathcal{H}_i)P(\mathcal{H}_i)}{P(D)}. \quad (8.3)$$

Notice that, assigning equal priors $P(\mathcal{H}_i)$ to the alternative models, **models \mathcal{H}_i are ranked by evaluating the evidence**, which naturally embodies the Occam's razor.

Given what we said so far, in order to perform model comparison it is necessary to evaluate what the Evidence is, i.e. we have to compute

$$P(D|\mathcal{H}_i) = \int P(D|\mathbf{w}, \mathcal{H}_i)P(\mathbf{w}|\mathcal{H}_i)d\mathbf{w}. \quad (8.4)$$

When the models we're dealing with are very complex, evaluating this integral can become infeasible. This notwithstanding, as the amount of data collected increases, a Gaussian approximation is expected to become increasingly accurate in describing our posterior. Moreover, if the distribution is very peaked we can perform a saddle-point approximation in Eq. (8.4), and write the Evidence as the product between the height of the peak of the integrand times its width, $\sigma_{w|D}$:

$$P(D|\mathcal{H}_i) \simeq \underbrace{P(D|\mathbf{w}_{MP}, \mathcal{H}_i)}_{\text{Best fit likelihood}} \times \underbrace{P(\mathbf{w}_{MP}|\mathcal{H}_i)}_{\text{Occam factor}} \sigma_{w|D} \quad (8.5)$$

Evidence \simeq Best fit likelihood \times Occam factor

Now suppose, for simplicity, that the prior $P(\mathbf{w}|\mathcal{H}_i)$ is uniform on some large interval σ_w , representing the range of values of \mathbf{w} that were possible *a priori*, according to \mathcal{H}_i (figure 8.3). Then, the **Occam factor** is equal to the ratio of the accessible volume of \mathcal{H}_i 's parameter space to the prior accessible volume

$$\text{Occam factor} = \frac{\sigma_{w|D}}{\sigma_w} \quad (8.6)$$

From this expression is manifest that the Occam factor penalizes either complex models having many parameters, each of which is free to vary over a large range σ_w , and models that have to be finely tuned to the data (small $\sigma_{w|D}$), favouring models for which the required precision of the parameters is coarse.

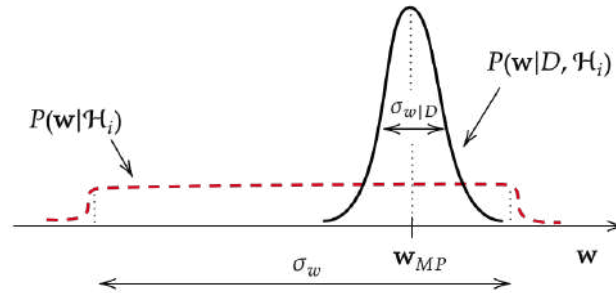


Figure 8.3: The prior distribution (dashed line) for the parameters has width σ_w . The posterior distribution (solid line) is peaked at w_{MP} and has a characteristic width $\sigma_{w|D}$. The Occam factor is the ratio between the two.

The Occam factor is therefore a measure of complexity of the model, and its logarithm represents the amount of information we gain about the model's parameters when the data arrive. Which model achieves the greatest evidence is determined by a trade-off by minimizing this natural complexity measure and minimizing the data misfit.

8.1.3 The big picture

Figure 8.4 summarizes all we said so far. We have three models $\mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3$, each of those has his prior probability distribution $P(w|\mathcal{H}_i)$. \mathcal{H}_3 is the most powerful/complex model, having a larger σ_w , whereas \mathcal{H}_1 is the simplest one. The clouds of points in the center represent the joint probability distributions of parameters and data given the specific model, $P(D, w, \mathcal{H}_i)$ (N.B. they are not data points! They are simulated data from the model.). Marginalizing this distribution with respect to the parameters, we get on the left the probability density that each model assigns over the data space, i.e. the evidence $P(D|\mathcal{H}_i)$.

The horizontal dashed line represents instead the specific data set we measured, D . As the data set is received, each prior distribution “collapses” in the corresponding posterior distribution $P(w|D, \mathcal{H}_i)$. In this case the simplest model \mathcal{H}_1 is not the one selected by inference, because the data fall far away from its data-prediction range and best-fit-likelihood term penalizes it a lot. Then, since the Occam factor $\sigma_{w|D}/\sigma_w$ is smaller for \mathcal{H}_3 than for \mathcal{H}_2 , given this data set, the most probable model is \mathcal{H}_2 .

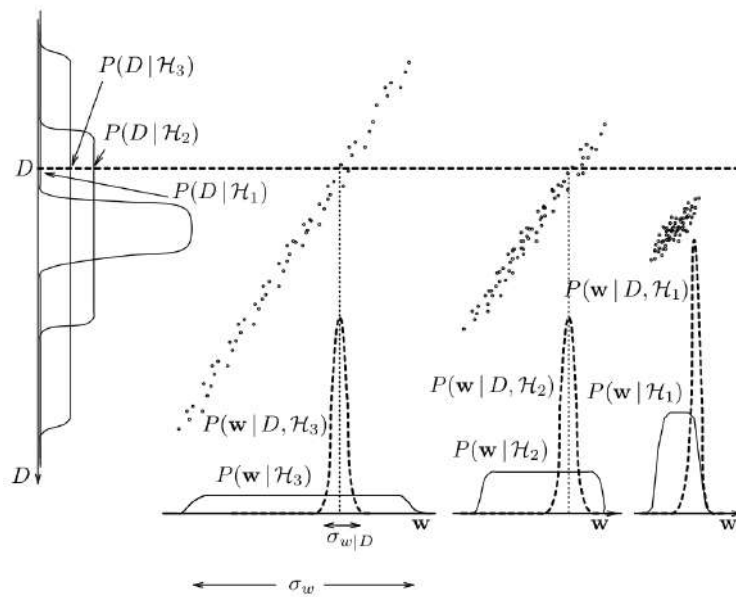


Figure 8.4: A hypothesis space consisting of three exclusive models, each having one parameter w and a one-dimensional data set D , the latter represented by the horizontal dashed line. The dots represent typical samples from the joint distribution $P(D, w, \mathcal{H})$ (they are not data points!). Figure taken from [Mac03], p. 350.

Exercise — 28.4, (Mac03)

The influence of race on the imposition of the death penalty for murder in America has been much studied. The following three-way table classifies 326 cases in which the defendant was convicted of murder. The three variables are the defendant's race, the victim's race, and whether the defendant was sentenced to death.

Quantify the evidence for the four alternative hypothesis:

- \mathcal{H}_{00} : The death penalty is independent of the skin's color of both victim (v) and defendant (m);
- \mathcal{H}_{01} : The death penalty depends on whether the defendant is black or white;
- \mathcal{H}_{10} : The death penalty depends on whether the victim is black or white;
- \mathcal{H}_{11} : The death penalty is influenced by both the skin's color of the victim and the defendant.

Solution: look at the end of this chapter. ■

	White defendant			Black defendant	
	Death penalty			Death penalty	
	yes	no		yes	no
White victim	19	132	White victim	11	52
Black victim	0	9	Black victim	6	97

8.2 Creating models, making choices and Bayesian inference

Let's make a pause and think about the picture reported in Fig. 8.6. We can ask ourselves *where is the parameter estimation and model comparison fit in the things?* At the beginning we have data, gathered in some experiments, and models. We fit each model in the data by making use of the tools we discussed in the parameter estimation section, and then we compare and assign preferences to the alternative models. The last part, also fundamental, it is often forgotten in the process: we **always ought to take decisions** after the model comparison phase, on whether we need more data (*What measurement should I do next that provides the best information?*), or create new models. At some point maybe we can also decide that we do not need

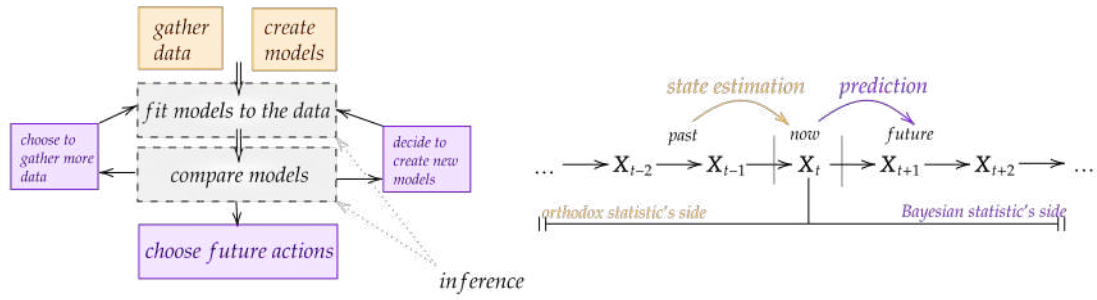


Figure 8.5: Where Bayesian inference fits into the data modelling process.

to gather more data or to change our model further. This kind of **feedback** is fundamental in the process. Data is sequential, and we gather it sequentially in time. In general, orthodox statistics concentrate its effort in data compression to represent the past, but the Bayesian approach is more like *what do I believe the data will be in the future?* The Bayesian approach is: *now I need to make a decision, and it is related to the prediction I can do.*

We can see in the same way what is happening here just by thinking that there are actions we can take in the world, for which we can have only a representation, and the latter spits out some data. Then we wonder how our actions are related to the data we saw, and this is the **model** part. Comparing models is key to figure out how actions in the data have anything to do with each other. We have the past, we can choose an action, and what we will see in the future depends on the action itself. Data is coming sequentially, and we need to be able to understand whether we need more data, we need to change the model or be satisfied and choose an action and see what happens.

So, now we suppose that we are fine with the model we ended up with: this is representing exactly a state that we think to be the state of the world, i.e. an inferred state of the world. Whatever the world is we cannot know it directly, but with our state we can take actions. The π is choosing the action to take in the world, and when it is effectively chosen we have a **way to do dynamics** with the T . In fact, with this arrow we can step forward into the future **with the model** and say, *what do I think the next state S_{t+1} will be?* Then we run f and declare “*I think my data will be \tilde{y}_{t+1}* ”. On the other side of the circle we **take effectively the action** through A and the world produces an outcome y_{t+1} . Now we compare the two. The result of this comparison is exactly what we use to evaluate our model. If \tilde{y}_{t+1} and y_{t+1} match, then the model is good. Pay attention to the distinction between what is physical and what is a representation, since we cannot really know what the world is, and we can only represent it. This is a sort of **Bayesian inference over time**. Using a control theory terminology, what we are trying to do is to maximize the mutual information we have with the world, or in other words we have a **policy π** that optimizes the mutual information between the actions and the observed data. We want to pick a policy π in order that it maximizes the following:

$$\operatorname{argmax}_{\pi} I(A, D) \quad (8.7)$$

In a nutshell: Suppose we are hiking on a forest, all it's fine, we are healthy and happy and at some point we see a plant with some berries: this is our actual state S_t . Now we can ask our self if they are good to eat or not; obviously we try to figure out in our mind what could happen if we eat some of those, i.e. we try making predictions T . At some point we decide to eat them, so we've taken an action A and then our body

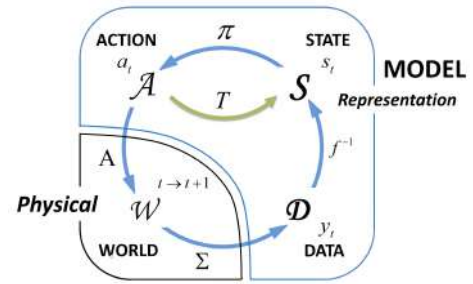


Figure 8.6: Where Bayesian inference fits into the data modelling process.

will react is some way, accordingly if those berries were poisonous or not. Now we collect data y_i from our body, like if we have fever or headache maybe caused by the berries and we try to infer our new health condition S_{t+1} . Here is where a collision could happen between our prediction T and what is our actual state S_{t+1} , if we were wrong, but at the end of the day we've learned if such berries are good to eat or not. This process is called **learning** i.e. go through the world, make predictions, be right and not be surprised of that.

Notice also the difference between **exploration** vs **exploitation**: the first one is when one goes through something never done before in order to learn predictions that work well. The second, instead, is when someone wants to go into the world, takes actions, but he doesn't want to learn something more. So there is a balance between them in the so called *exploration-exploitation problem*, where one wants to learn something and then apply what he has learned.

From the mathematical point of view we have:

- Model State: $p(s_t)$
- Dynamics Model: $T = p(s_{t+1}|a_t, s_t)$
- Observation Model: $f = p(y_t|s_t)$
- Policy: $a_t = \pi[p(s_t)]$ that is not really a probability but rather a decision-action.

In the simplest case when there is no policy, Fig. 8.6 is reduced to:

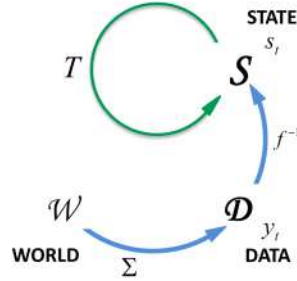


Figure 8.7: Dynamical System with no policy.

We can write the state estimation as

$$p(s_t|y_t, a_{t-1}) = \sum_{s_{t-1} \in \mathbf{S}} p(s_t|y_t, a_{t-1}, s_{t-1})p(s_{t-1})$$

where

$$p(s_t|y_t, a_{t-1}, s_{t-1}) = \frac{p(y_t|s_t)p(s_t|a_{t-1}, s_{t-1})}{p(y_t|a_{t-1}, s_{t-1})} = \frac{p(y_t|s_t)p(s_t|a_{t-1}, s_{t-1})}{\sum_{s_{t+1} \in \mathbf{S}} p(y_{t+1}|s_{t+1})p(s_{t+1}|a_t, s_t)}$$

As we are going to see later on, this problem can be seen under the perspective of the *information bottleneck*. The goal is to compress the past as much as possible while retaining the information relevant for predicting the future, formally solving

$$p^*(s_{t+1}|a_t) = \underset{p(s_{t+1}|a_t)}{\operatorname{argmin}} \underbrace{I(A, S)}_{\text{Compression}} - \beta \underbrace{I(S, Y)}_{\text{Relevance}} = \underset{p(s_{t+1}|a_t)}{\operatorname{argmin}} \mathcal{F}[p(s_{t+1}|a_t)] \quad (8.8)$$

The formal solution is

$$p(s_{t+1}|a_t) = \frac{p(s_{t+1})}{Z(a_t, \beta)} \exp \left[-\beta \sum_{y \in \mathcal{Y}} p(y_{t+1}|a_t) \log \frac{p(y_{t+1}|a_t)}{p(y_{t+1}|s_{t+1})} \right]$$

where

$$p(y_{t+1}|s_{t+1}) = \frac{1}{p(s_{t+1})} \sum_{a \in \mathcal{A}} p(y_{t+1}|a_t) p(s_{t+1}|a_t) p(a_t)$$

and

$$p(s_{t+1}) = \sum_{a \in \mathcal{A}} p(s_{t+1}|a_t) p(a_t)$$

Please notice that in the simplest case with no policy the presence of a_t and a_{t-1} disappear from the previous equations.

8.3 Language model and distance in the space of parameters

Suppose we want to build a language model, and we want to measure the distance between two languages: English and Spanish. We need a metric in the space of models for this kind of measurements. In machine learning this is the point where a loss function comes into play, but here we will try to use KL divergence instead.

Let's take into account the **Morse code** reported in Fig. 8.8, which is a method used in telecommunication to encode text characters as standardized sequences of two different signal's durations, called dots and dashes. To increase the efficiency of encoding, Morse code was designed so that the length of each symbol is approximately **inverse to the frequency of occurrence** of the character that it represents in text of the English language (note the letter *e* as the most present in the common English language). So, it was optimized for the English language, and if we look at the following probability distributions of occurrence (Fig. 8.9) of single letters p and Spanish q respectively, we can notice that they are substantially different. *How KL works for probability models we would like to compare?* Suppose we now build the Morse code for the Spanish language, using the Spanish frequencies, but we still want to encode English through this. In a sense, **this is not an optimal thing to do**, just because the two distributions are different.

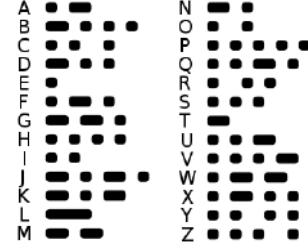


Figure 8.8: The Morse code.

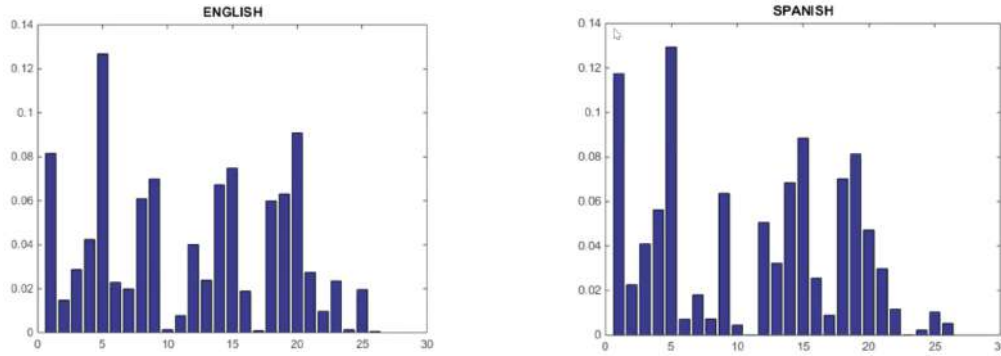


Figure 8.9: Probabilities of single letters A-Z in English p and in Spanish q respectively, the two distributions are quite different.

The question is: *How many more bits do I need to use per letter if I was talking in English but I did Morse encoding with the Spanish distribution?* The answer is given by the KL-divergence:

$$D_{KL}(p||q) = \sum_{i=1}^{26} p_i \log \frac{p_i}{q_i} = 0.37 \text{ bits/letter}$$

$$D_{KL}(q||p) = \sum_{i=1}^{26} q_i \log \frac{q_i}{p_i} = 0.19 \text{ bits/letter}$$

This must be read like: if we want to encode an English sentence in a Morse code optimized for the Spanish language we need to add, on average, 0.37 bits per letter. And notice again how **this is not a distance**: it is more optimal to encode Spanish with an English probability model rather than the opposite. English is further from Spanish than Spanish is further from English. If we need to define a measure we find some failures with the KD divergence, namely it does not obey the **reflexive property** and the **triangular inequality**. We need to make a step further, thinking about an enormously complex and high dimensional space (figure 8.10), where models live. The English and Spanish models are sitting in some places p and q . There is no way to compare projections in the tangent space, i.e. there is no way to compare different models with an euclidean metrics, because the space itself is not Euclidean! The thing we want to do is to measure the real distance between the blue and red (dotted line). Space can be curve in high dimension and we need to form a more general mathematical structure than a Euclidean vector space, by **allowing the inner product to be a function of the coordinates**. We call ds the differential distance between objects in the space:

$$(ds)^2 = d\mathbf{x}^T g(\mathbf{x}) d\mathbf{x}$$

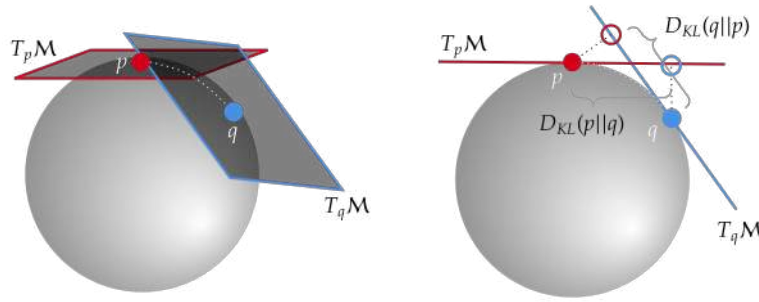


Figure 8.10: The probability models's space. The red empty circle is like *where is the Spanish projected into the English space?*

where $g(\mathbf{x})$ is a spatially varying **metric tensor**. In our case we are in the space of probability distributions and, denoting $\boldsymbol{\omega}$ the parameters of the distributions, we have that the infinitesimal arc length is given by

$$(ds)^2 = d\boldsymbol{\omega}^T g(\boldsymbol{\omega}) d\boldsymbol{\omega}.$$

For example, we could be in the space of Gaussians defined by μ and σ^2 , so $\boldsymbol{\omega} = (\mu, \sigma^2)$. Conjugate priors are convenient, and this is exactly the space in which we are moving around when we update the parameters after inference. To sum up, the key point is that **information-theoretic measures induce a geometry on the space of probability models** (the Statistical Manifold).

We start with the KL divergence

$$D_{KL}(p' || p) = \int d\mathbf{x} p(\mathbf{x} | \boldsymbol{\omega}) \log \frac{p(\mathbf{x} | \boldsymbol{\omega}')}{p(\mathbf{x} | \boldsymbol{\omega})},$$

where the prime is put simply to indicate that is a different set of parameters, and we expand $p(\mathbf{x} | \boldsymbol{\omega})$ around $\boldsymbol{\omega}$. In other words, we are in a point of the curve $\boldsymbol{\omega}$ and we want to find which is the tangent plane passing through the point by infinitesimally changing the parameters. We obtain:

$$D_{KL}(p(\mathbf{x} | \boldsymbol{\omega}') || p(\mathbf{x} | \boldsymbol{\omega})) = \frac{1}{2} (\boldsymbol{\omega}' - \boldsymbol{\omega})^T G(\boldsymbol{\omega}) (\boldsymbol{\omega}' - \boldsymbol{\omega}) + O(|\boldsymbol{\omega}' - \boldsymbol{\omega}|^3)$$

where $G(\boldsymbol{\omega})$ is the **Fisher information matrix**

$$G_{\alpha\beta}(\boldsymbol{\omega}) = \int d\mathbf{x} p(\mathbf{x} | \boldsymbol{\omega}) \frac{\partial \ln p(\mathbf{x} | \boldsymbol{\omega})}{\partial \omega^\alpha} \cdot \frac{\partial \ln p(\mathbf{x} | \boldsymbol{\omega})}{\partial \omega^\beta} = - \int d\mathbf{x} p(\mathbf{x} | \boldsymbol{\omega}) \frac{\partial^2 \ln p(\mathbf{x} | \boldsymbol{\omega})}{\partial \omega^\alpha \partial \omega^\beta} \quad (8.9)$$

that is exactly the metric tensor for probability models.

8.3.1 Using Fisher information: the Fisher scoring algorithm

The Fisher information matrix has many uses in complexity views of probabilistic modelling. It quantifies the “resolving” power of a measurement to determine the parameters of statistical model, as we describe in this section. Given the data $\mathbf{x} \in \mathbb{R}^d$ and the model $p(\mathbf{x} | \boldsymbol{\omega})$, we can define

$$\mathbf{U}(\boldsymbol{\omega} | \mathbf{x}) = \nabla_{\boldsymbol{\omega}} \ln p(\mathbf{x} | \boldsymbol{\omega}) \quad \text{Fisher score}$$

$$G(\boldsymbol{\omega}) = \langle \mathbf{U} \mathbf{U}^T \rangle_x = \int d\mathbf{x} p(\mathbf{x} | \boldsymbol{\omega}) \mathbf{U}(\boldsymbol{\omega} | \mathbf{x}) \mathbf{U}^T(\boldsymbol{\omega} | \mathbf{x}) \quad \text{Expected Fisher information}$$

$$\begin{aligned} G_{emp}(\boldsymbol{\omega}) &= \langle \mathbf{U} \mathbf{U}^T \rangle_{emp} = \int d\mathbf{x} p_{emp}(\mathbf{x}) \mathbf{U}(\boldsymbol{\omega} | \mathbf{x}) \mathbf{U}^T(\boldsymbol{\omega} | \mathbf{x}) \\ &\stackrel{*}{=} \frac{1}{N} \sum_{i=1}^N \mathbf{U}(\boldsymbol{\omega} | \mathbf{x}_i) \mathbf{U}^T(\boldsymbol{\omega} | \mathbf{x}_i) \quad \text{Observed Fisher information} \end{aligned}$$

where in $(*)$ we substituted the empirical measure

$$p_{emp}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \delta(\mathbf{x} - \mathbf{x}_i)$$

Suppose that we want to perform a Maximum Likelihood Estimation (MLE) of our parameters. The target function that we want to maximize with respect to $\boldsymbol{\omega}$ is the (rescaled) log-likelihood

$$L(\boldsymbol{\omega}) \equiv \frac{1}{N} \sum_{i=1}^N \ln p(\mathbf{x}_i | \boldsymbol{\omega}) = \langle \ln p(\mathbf{x} | \boldsymbol{\omega}) \rangle_{emp} \quad (8.10)$$

To this end, let us consider the second order Taylor expansion of the latter around $\boldsymbol{\omega}$

$$L(\boldsymbol{\omega} + \delta\boldsymbol{\omega}) \approx L(\boldsymbol{\omega}) + \nabla L(\boldsymbol{\omega}) \delta\boldsymbol{\omega} + \frac{1}{2} \delta\boldsymbol{\omega}^T \nabla \nabla L(\boldsymbol{\omega}) \delta\boldsymbol{\omega}. \quad (8.11)$$

To maximize this quantity, we shall take the gradient with respect to $\delta\boldsymbol{\omega}$ and compute it at the optimal point $\delta\boldsymbol{\omega}_{opt}$. Since at the stationary point we have $\nabla L(\boldsymbol{\omega} + \delta\boldsymbol{\omega}_{opt}) = 0$, we're left with

$$\mathbf{0} = \nabla L(\boldsymbol{\omega}) + \nabla \nabla L(\boldsymbol{\omega}) \delta\boldsymbol{\omega}_{opt} \quad (8.12)$$

Once we set $\delta\boldsymbol{\omega}_{opt} = \boldsymbol{\omega}^{t+1} - \boldsymbol{\omega}^t$, we can think to maximize the target function by iteratively update our parameters as

$$\boldsymbol{\omega}^{t+1} = \boldsymbol{\omega}^t - [\nabla \nabla L(\boldsymbol{\omega}^t)]^{-1} \nabla L(\boldsymbol{\omega}^t) \quad (8.13)$$

which is called *Newton's algorithm*. In our specific case, the Hessian in the previous formula is nothing but the observed Fisher information matrix. If we also introduce a tunable learning rate η , we get the so called **Fisher's scoring algorithm**

$$\boldsymbol{\omega}^{t+1} = \boldsymbol{\omega}^t + \eta [G_{emp}(\boldsymbol{\omega}^t)]^{-1} \langle \nabla_{\boldsymbol{\omega}} \ln p(\mathbf{x} | \boldsymbol{\omega}) \rangle_{emp} \quad (8.14)$$

Substituting the expected Fisher information for the observed one can help to stabilize the update for high-dimensional problems, since the data may not provide enough regularization to accurately estimate the matrix.

8.4 Information Geometry

In the previous sections we pointed out the fact that the space of probability models is not a vector space, but a non-trivial manifold in which we can define a metric tensor $g(\boldsymbol{\omega})$ that allows to compute the infinitesimal arc length Lesson 14
17/05
AZ

$$(ds)^2 = d\boldsymbol{\omega}^T g(\boldsymbol{\omega}) d\boldsymbol{\omega} \quad (8.15)$$

Roughly speaking, the *length* of a path connecting two models in the manifold is given by the integral over the path of the metric tensor; the *distance* is defined as the minimum of those possible length, and the corresponding path is called **geodesic**.

The statistical manifold is made of all models belonging to a specific family of distributions (for example the family of 1d Gaussian pdfs $\mathcal{N}(\mu, \sigma)$), and hence moving from $A \rightarrow B$ in the statistical manifold means that we are moving in the parameters' space of such distributions (for the Gaussian, it is the plane (μ, σ)). But again, this is not an Euclidean space: the geodesic connecting two points is not like a straight line.

But why the statistical manifold is not an euclidean space? The easiest answer is that in such space there are paths that are unfeasible. For example, think about the travel between Hawaii and South-Africa: from an Euclidean point of view the geodesic is just the line connecting such points, but this would imply to travel through the core of the Earth. So, if we require to stay in space of "realizable travels", the price to pay is to adopt a more complicated math. Here is indeed where Information Theory and General Relativity touch each other.

8.4.1 Riemannian Manifold

Let start first with some definitions in order to set up the theoretical framework.

Definition 8.1 A **Manifold** is a topological space that is locally Euclidean (i.e., around every point, there is a neighborhood that is topologically the same as the open unit ball in \mathbb{R}^n).

Definition 8.2 Let M be a smooth manifold. A **Riemannian metric** g on M is a smooth family of inner products on the tangent spaces of M . Namely, g associates to each $p \in M$ a positive definite symmetric bilinear form on $T_p M$, i.e.

$$g_p : T_p M \times T_p M \rightarrow \mathbb{R}$$

Definition 8.3 A **Riemannian Manifold** is a pair (M, g) where M is a differentiable manifold and g is a Riemannian metric on M .

How can we find the geodesics on a Riemannian Manifold? The idea is to consider all the paths connecting two points A and B as trajectories followed by a dynamical process. Suppose to have n parameters characterizing the model, and a submersion describing the manifold $\tilde{X} : \mathbb{R}^n \mapsto M$. For a given time interval $[s_A, s_B]$, we can associate the curve described by the parameters in the Euclidean space

$$\mathbf{q} : [s_A, s_B] \mapsto \mathbb{R}^n$$

Once we map these curves on the manifold, we can define the functional *length* as

$$\tilde{l}[\tilde{\mathbf{q}}] \equiv \int_{s_A}^{s_B} \|\tilde{\mathbf{q}}'(s)\| ds, \quad \tilde{\mathbf{q}} = \tilde{X} \circ \mathbf{q}$$

and since

$$\|\tilde{\mathbf{q}}'(s)\|^2 = [\tilde{X}'(\mathbf{q}(s))\mathbf{q}'(s)]^T \tilde{X}'(\mathbf{q}(s))\mathbf{q}'(s) = \mathbf{q}'(s)^T g(\mathbf{q}(s)) \mathbf{q}'(s), \quad g(\mathbf{q}) \equiv \tilde{X}'(\mathbf{q})^T \tilde{X}'(\mathbf{q}) \in \mathcal{M}_{n \times n}$$

we can equivalently write

$$\tilde{l}[\tilde{\mathbf{q}}] = l[\mathbf{q}] \equiv \int_{s_A}^{s_B} \sqrt{\mathbf{q}'(s)^T g(\mathbf{q}(s)) \mathbf{q}'(s)} ds$$

and the geodesic is the curve $\mathbf{q}(s)$ that maximizes this functional. It turns out that this problem is equivalent to minimize the *Energy functional*

$$E[\mathbf{q}] \equiv \frac{1}{2} \int_{s_A}^{s_B} \mathbf{q}'(s)^T g(\mathbf{q}(s)) \mathbf{q}'(s) ds = \int_{s_A}^{s_B} L(\mathbf{q}(s), \mathbf{q}'(s)) ds$$

where we recognized a purely kinetic Lagrangian in which the metric tensor plays the role of “the inverse of a mass” $1/m$. There’s no need of a potential term that constrains the trajectories on the manifold: all the information about the geometry of the manifold is already embodied in the metric tensor.

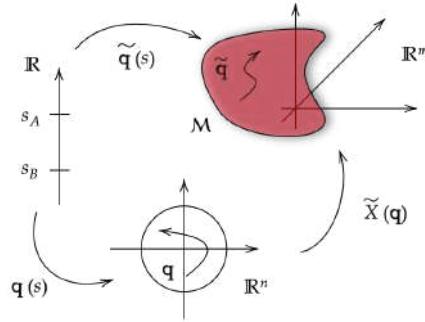


Figure 8.11: Representation of the mapping between the time, the Euclidean space of parameters and the statistical manifold

Again, all this construction is made possible by of conjugate priors, which allow to keep the trajectory on the same statistical manifold. The data we collect act like a force that pushes the dynamics from one point to another on the manifold, and the length of the path in the model space is therefore the information content that we’ve learnt during the measurement, i.e. the length of the geodesic.

The stationarity condition for the energy functional, $\delta E = 0$, is satisfied for those trajectories that satisfy the Euler-Lagrange equations

$$\frac{\delta L}{\delta q^k} = \frac{\partial L}{\partial q^k} - \frac{d}{ds} \frac{\partial L}{\partial \dot{q}^k} = 0 \quad \text{where} \quad \frac{d}{ds} = \frac{\partial}{\partial s} + \dot{q}^l \frac{\partial}{\partial q^l} + \ddot{q}^l \frac{\partial}{\partial \dot{q}^l}$$

In the end one finds a set of ODEs that, if solved, gives all the possible geodesics

$$\ddot{q}^k + \dot{q}^i \Gamma_{ij}^k \dot{q}^j = 0 \quad (8.16)$$

where Γ_{ij}^k is the Christoffel symbol

$$\Gamma_{ij}^k = \frac{1}{2} g^{km} (\partial_i g_{mj} + \partial_j g_{im} - \partial_m g_{ij})$$

8.4.2 Example: 1-D Gaussian pdf

In order to picture the abstract theory discussed above, let's start with the paradigmatic example of the Gaussian pdf. In this case the set of parameters is $\boldsymbol{\omega} = (\mu, \sigma)$

$$p(x|\boldsymbol{\omega}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$$

By changing these two parameters, obviously we expect Gaussians with different shapes and positions. The metric tensor is given by the Fisher information

$$g_{ij}(\boldsymbol{\omega}) = - \int d\mathbf{x} p(\mathbf{x}|\boldsymbol{\omega}) \frac{\partial^2 \ln p(\mathbf{x}|\boldsymbol{\omega})}{\partial \omega_i \partial \omega_j} = \frac{1}{\sigma^2} \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$$

where the indices i, j span through all possible Gaussians: $\boldsymbol{\omega}_i = (\mu_i, \sigma_i) \in \mathbb{R} \times \mathbb{R}^+$. A nice representation of what is going on is presented in Fig. 8.12. The upper plot is the usual data space, while the other one

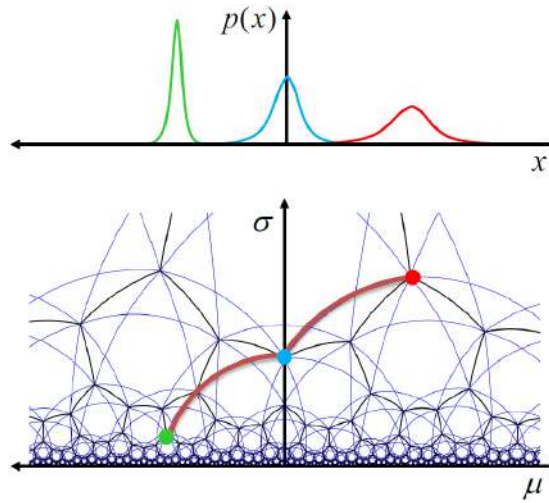


Figure 8.12: Data space topology vs model space topology.

shows the hyperbolic geometry of the Poincaré upper-half plane, \mathcal{H}^2 , of the models' space. The red lines connecting the three points correspond to geodesics between the different pdfs.

Using Eq. 8.15 we can write

$$(ds)^2 = \frac{1}{\sigma^2} (d\mu^2 + 2d\sigma^2).$$

Notice the key point here: σ is at the denominator of the infinitesimal arc length, meaning that we can have a longer distance if we move in a vertical line from top to bottom, and indeed when $\sigma \rightarrow 0$ then $ds \rightarrow \infty$. The interpretation is pretty nice, because when $\sigma \rightarrow 0$, $p(x|\mu, \sigma) \rightarrow \delta(\mu)$, and naively it is reasonable that the distance between delta functions is infinite.

Finally, the analytic form of the real distance in the model space of 1-D Gaussians is obtained solving Eq. (8.16), and it is

$$r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_j) = \sqrt{2} \cosh^{-1} \left(1 + \frac{(\mu_i - \mu_j)^2 + 2(\sigma_i - \sigma_j)^2}{4\sigma_i \sigma_j} \right).$$

Remember that in Eq. (8.9) we used the natural logarithm, so the unit measure of such distance $r(\omega_i, \omega_j)$ is *nats*; if instead we used the base-2 logarithm it would have been measured in *bits*. Some numerical results are shown in Fig. 8.13.

Proposition 8.1 If $\delta = |\mu_i - \mu_j| \ll \sigma = \varepsilon$ then

$$r \approx \frac{\delta}{\varepsilon}$$

i.e. the distance between "local" high precision data is Euclidean.

Proof.

$$\begin{aligned} r(\omega_i, \omega_j) &= r\left((\mu_i, \sigma_i = \varepsilon), (\mu_j = \mu_i + \delta, \sigma_j = \varepsilon)\right) \\ &= \sqrt{2} \cosh^{-1}\left(1 + \frac{\delta^2}{4\varepsilon^2}\right) = \sqrt{2} \ln\left[1 + \frac{\delta^2}{4\varepsilon^2} + \sqrt{\left(1 + \frac{\delta^2}{4\varepsilon^2}\right)^2 - 1}\right] \\ &= \sqrt{2} \ln\left[1 + \frac{\delta^2}{4\varepsilon^2} + \frac{\delta}{\sqrt{2}\varepsilon} \sqrt{1 + \frac{\delta^2}{8\varepsilon^2}}\right] \approx \frac{\delta}{\varepsilon} \end{aligned}$$

■

The big deal of this approach is that a distributed object, like a pdf, becomes a point in the right space, which in our case is the models' space. Since there are many more techniques and tools to handle points than pdfs, this idea paves the way to a lot of possible applications. The price to pay is that the space of such points is not Euclidean anymore; indeed, as shown in Fig. 8.13, $\hat{A}\hat{B}\hat{C}$ forms a really weird isosceles triangle in a negative curvature space. Recall that the sum of the angles of a triangle is:

- $\alpha + \beta + \gamma \leq \pi$ if the space has negative curvature;
- $\alpha + \beta + \gamma = \pi$ if the space has null curvature (Euclidean case);
- $\alpha + \beta + \gamma \geq \pi$ if the space has positive curvature.

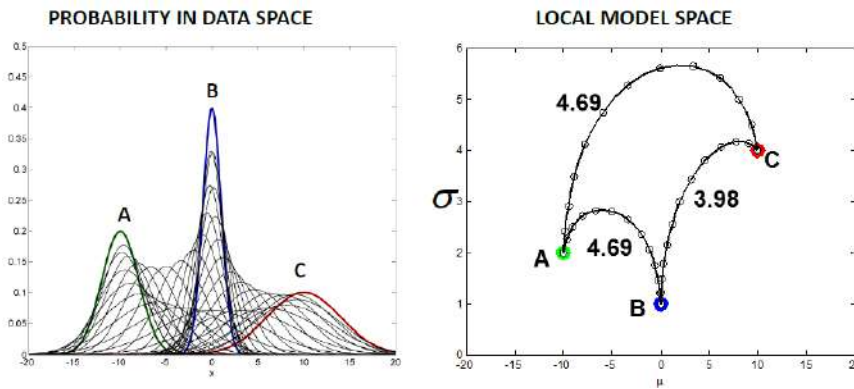


Figure 8.13: Space comparison between 1-D gaussian.

Moreover, this real distance r is invariant under change of coordinates, i.e. the geodesics' shape changes if we redefine the parameters used to describe our pdf, but its length doesn't. In figure 8.14 are shown the most used parametrizations for a Gaussian.

Obviously all things said up to now can be easily generalized with multidimensional pdfs, just everything becomes more cumbersome. For example, if we want to study 2-D Gaussians, by the fact that they are described by 5 parameters $\omega = (\mu_x, \mu_y, \sigma_x, \sigma_{xy}, \sigma_y)$, we need a 5-dim statistical manifold. So if we follow a 5-D geodesic in model space and we project it in a 2-D data space, we end up with figures like the one on the left of figure 8.15. In the case of 3-D Gaussian pdfs instead, we get figures like the one on the right.

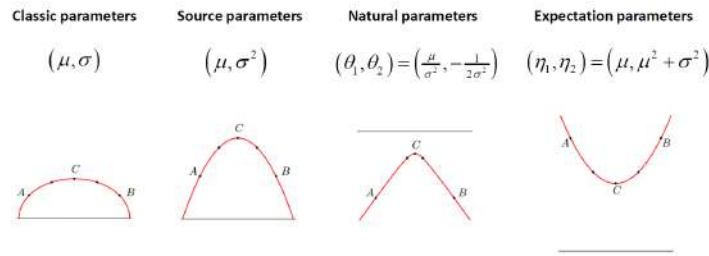


Figure 8.14: Geodesic's shape for different parametrizations.

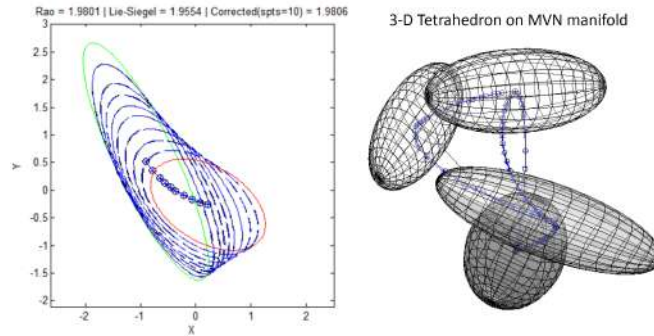


Figure 8.15: Level curves for 2-D and 3-D Gaussians.

8.4.3 Connection with Inference

Now that we've seen all the technicalities behind information geometry, we shall concentrate on the meaning of that in terms of inference. Up to now we've seen what it's called *parametric inference approach*, i.e. we select a parametric probability distribution $p(\mathbf{x}|\boldsymbol{\omega})$, the likelihood, and, given the data, all what we have to do is to find $\boldsymbol{\omega}_{est}$. This is much easier compared to the non-parametric approach, where we don't even make any assumption on the pdf underneath the data, but it's quite limited. Sometimes an approach that lays in the middle of the two is desirable.

For example, think about what happens if we have a non-Gaussian, multi-modal pdf. A good idea in this case can be to approximate the pdf by means of a *mixture of Gaussians* $p(\mathbf{x}|\boldsymbol{\omega})$, hence following an hybrid approach based on information geometry called **statistical manifold density estimation**. Here we can estimate $p(\mathbf{x})$ (the real probability density of the data present in the world) with a $q(\mathbf{x})$ that comes from our model defined as

$$q(\mathbf{x}) = \int d\boldsymbol{\omega} p(\mathbf{x}|\boldsymbol{\omega}) \hat{\pi}(\boldsymbol{\omega}). \quad (8.17)$$

Here, $\hat{\pi}(\boldsymbol{\omega})$ is a pdf on the statistical manifold that is expected to be an estimate of the true $\pi(\boldsymbol{\omega})$ out there in the world. We proceed in such a way with the hope that the estimate of $\hat{\pi}(\boldsymbol{\omega})$ in the manifold is an easier task compared to the direct estimation of $p(\mathbf{x})$.

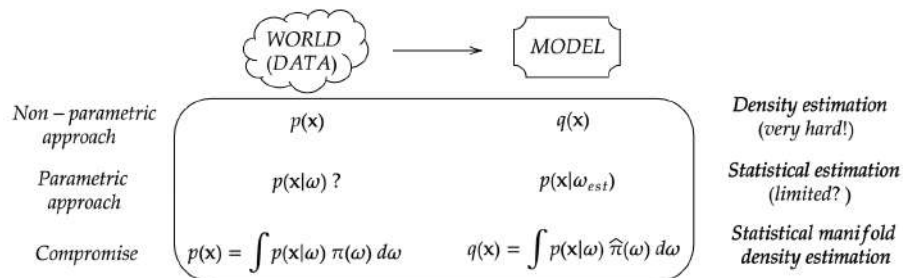


Figure 8.16: Different approaches to inference

This method of using a Gaussian basis in order to approximate any arbitrarily complex $p(x)$ with $q(x)$ is also called **Gaussian Mixture Model (GMM)**. In the discrete case we have

$$p_K(\mathbf{x}) = \sum_{k=1}^K \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \pi_k \quad \text{where} \quad \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = |2\pi\boldsymbol{\Sigma}_k|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)\right\}$$

then, if we take the continuum limit, we obtain the so called **cGMM**

$$p(\mathbf{x}) = \lim_{K \rightarrow \infty} \lim_{\det\{\boldsymbol{\Sigma}_k\} \rightarrow 0} p_K(\mathbf{x}) = \int d\boldsymbol{\omega} \mathcal{N}(\mathbf{x}|\boldsymbol{\omega}) \pi(\boldsymbol{\omega})$$

where

$$(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \boldsymbol{\omega} \in \mathbb{M} = \mathbb{R}^d \times \text{Sym}_+(d, \mathbb{R}) \quad \text{and} \quad d\boldsymbol{\omega} = \sqrt{\det|g(\boldsymbol{\omega})|} \prod_{i=1}^n d\omega_i$$

This means that a finite sample on the cGMM manifold recovers the discrete GMM

$$\pi(\boldsymbol{\omega}) = \sum_{k=1}^K \pi_k \frac{\delta(\boldsymbol{\omega} - \boldsymbol{\omega}_k)}{\sqrt{\det|g(\boldsymbol{\omega}_k)|}}$$

In a nutshell: In the same way Fourier Transform is the main tool to rewrite any signal $f(x)$ in terms of periodic functions, cGMM can express any pdf $p(x)$ in terms of Gaussians.

$$f(\mathbf{x}) = \int_{-\infty}^{\infty} d\mathbf{k} u(\mathbf{x}|\mathbf{k}) a(\mathbf{k}) \Leftrightarrow p(\mathbf{x}) = \int d\boldsymbol{\omega} \mathcal{N}(\mathbf{x}|\boldsymbol{\omega}) \pi(\boldsymbol{\omega})$$

Summing up, we've seen that there is a strong and deep connection between probability and geometry that could be visualized as follows

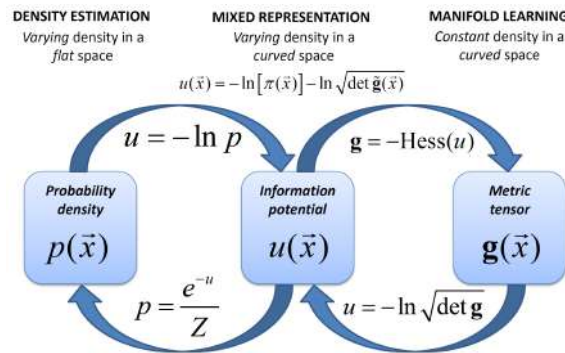


Figure 8.17: Relationship between probability and geometry.

This interpretation opens the doors on how we could tackle the problem of clustering. In general, clustering in unsupervised learning is a challenging problem, because although there are different algorithms on the market, all of them have some weaknesses.

Information geometry can really help us in this task. By looking at figure 8.18, the main idea behind is that we don't want anymore to find the right cluster shapes in our data space (A), but rather we aim to find a manifold where data are uniform (C) or, and this is indeed the smartest choice, to find cluster shapes on a manifold (B).

Let us make an example to put things in perspective. Suppose that we have a robotic hand that holds and manipulates a cube, and we want to recognize some common feature of the possible movements through a clustering technique. In this case, the space (C) represents the space of all the possible configurations of the hand, i.e. all the possible states allowed by its mechanical construction. Since we are considering a particular task though, that is manipulating a cube, only a subspace of the whole space will be explored by the process. For instance, those configurations in which the hand is open and upside down do not have to be taken into account. This restricted space is what we represented with (B) in the figure, and here the configurations tend to gather naturally in clusters that can be spotted with a proper algorithm.

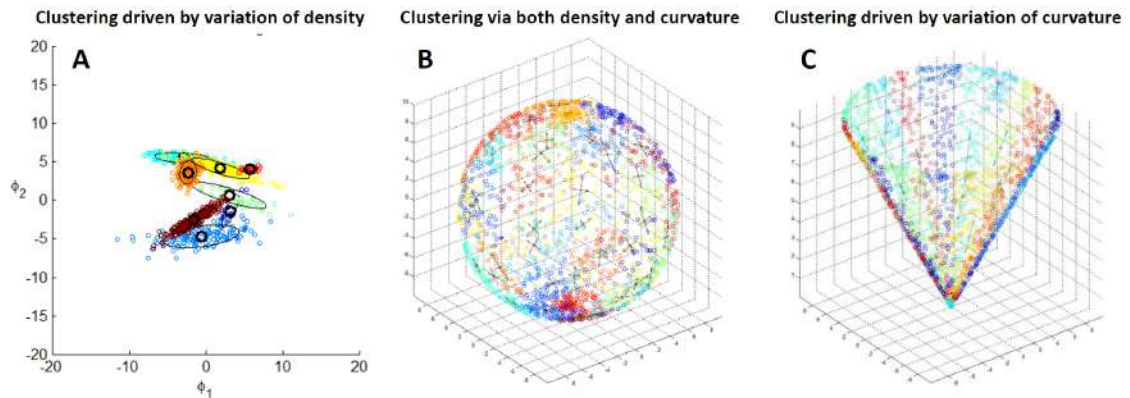


Figure 8.18: Clustering on Manifold.

8.5 Exercise solutions

8.5.1 Exercise 28.4 (Mac03)

For the sake of notation, let us map all the possible occurrences for the death penalty d , the victim v and the defendant m using binary variables: 0 if white and 1 if black.

m	White defendant 0		Black defendant 1			
	Death penalty		Death penalty			
d	yes 1	no 0	yes 1	no 0		
	White victim 0	19	132	White victim 0	11	52
v	Black victim 1	0	9	Black victim 1	6	97

Then, we can label each entry of the table above as

$$n_{dvm} \begin{cases} n_{100} = 19 & n_{000} = 132 & n_{101} = 11 & n_{001} = 52 \\ n_{110} = 0 & n_{010} = 9 & n_{111} = 6 & n_{011} = 97 \end{cases}$$

Figure 8.19 shows the probabilistic graphical models (PGM) for the four hypothesis, whereas figure 8.20 represents a schematic way of thinking to the various problems. Let us consider each case separately.

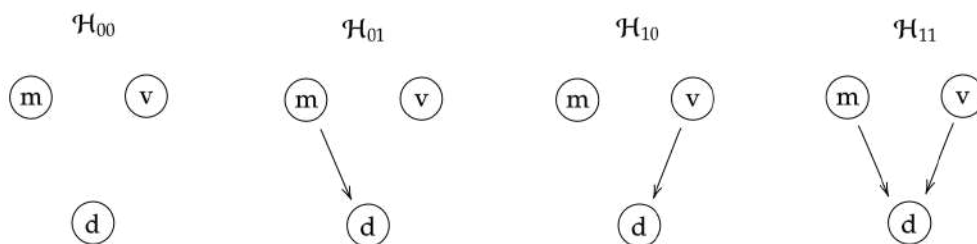


Figure 8.19: Probabilistic graphical models (PGM) for the four hypothesis

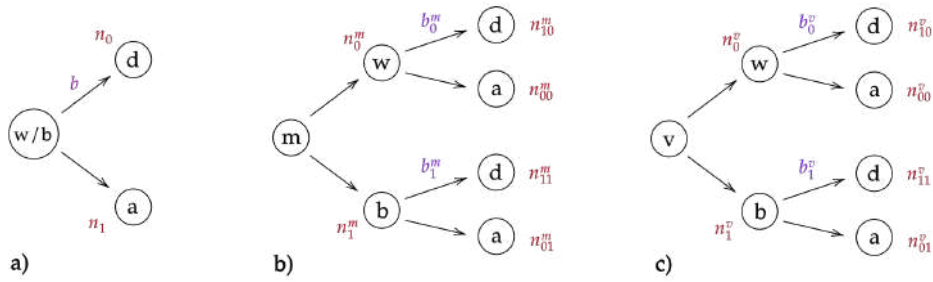


Figure 8.20: Schematic representation of the different problems. The meaning of the labels in the nodes is: m = defendant, v = victim, w = white, b = black, d = dead, a = alive

\mathcal{H}_{00}

In the simplest hypothesis the probability for the defendant to be sentenced to death is not affected by any racial prejudice. In this case, the model will have just one parameter, b , which represents the probability of death. The problem is therefore perfectly equivalent to the tossing of a coin having a bias b , as we represented in figure 8.20, a). The data for this model is made of the number of people sentenced to death and the total number of people

$$n_1 = \sum_{v=0}^1 \sum_{m=0}^1 n_{1vm} = 19 + 0 + 11 + 6 = 36$$

$$N = \sum_{d=0}^1 \sum_{v=0}^1 \sum_{m=0}^1 n_{dvm} = 326$$

The posterior distribution for the bias is

$$p(b|n_1, N; \mathcal{H}_{00}) = \frac{p(n_1|b, N; \mathcal{H}_{00})p(b|\mathcal{H}_{00})}{p(n_1|\mathcal{H}_{00})},$$

where the likelihood is a binomial distribution and for the prior we can choose a Beta distribution with parameters α and β . From the coin tossing problem we know that the evidence takes the analytical form

$$p(n_1|\mathcal{H}_{00}) = \binom{N}{n_1} \frac{B(\alpha + n_1, \beta + N - n_1)}{B(\alpha, \beta)}$$

\mathcal{H}_{01}

In this second model we assume that the death penalty's occurrence depends whether the *defendant*, m, is black or white. Now things start to become a little bit tricky, and we have to identify what is the question of the problem. The PGM of figure 8.19 reads: for each trial, what is the probability of death *given that* the defendant is black/white? A key observation is that the probability for a black person to be sentenced to death is independent of the probability that a white person is sentenced to death: even if there was a bias, the two processes are still independent from each other. This time it's like having two coins, the first with bias b_0^m and the second with bias b_1^m , and seeing how many heads and tails we get by tossing the former n_0^m times (the number of white defendant) and the latter n_1^m (the number of black defendant), as we depicted in figure 8.20, b). The model is therefore described by two parameters, namely the two biases.

The data for this problem are:

$$n_0^m = \sum_{d=0}^1 \sum_{v=0}^1 n_{dv0} = 19 + 132 + 0 + 9 = 160 \quad \text{White defendant}$$

$$n_1^m = \sum_{d=0}^1 \sum_{v=0}^1 n_{dv1} = 11 + 52 + 6 + 97 = 166 \quad \text{Black defendant}$$

$$n_{10}^m = \sum_{v=0}^1 n_{1v0} = 19 + 0 = 19 \quad \text{Sentenced to death, white defendant}$$

$$n_{11}^m = \sum_{v=0}^1 n_{1v1} = 11 + 6 = 17 \quad \text{Sentenced to death, black defendant}$$

The likelihood for the problem is

$$p(n_{10}^m, n_{11}^m | n_0^m, n_1^m, b_0^m, b_1^m; \mathcal{H}_{01}) = \prod_{j=0}^1 p(n_{1j}^m | n_j^m, b_j^m) = \text{Binom}(n_{10}^m | n_0^m, b_0^m) \times \text{Binom}(n_{11}^m | n_1^m, b_1^m)$$

and if we choose the same Beta conjugate prior with parameters α and β for both the parameters, we end up with an evidence

$$p(n_{10}^m, n_{11}^m, n_0^m, n_1^m | \mathcal{H}_{01}) = \binom{n_0^m}{n_{10}^m} \frac{B(\alpha + n_{10}^m, \beta + n_0^m - n_{10}^m)}{B(\alpha, \beta)} \times \binom{n_1^m}{n_{11}^m} \frac{B(\alpha + n_{11}^m, \beta + n_1^m - n_{11}^m)}{B(\alpha, \beta)}$$

\mathcal{H}_{01}

The very same argument of the previous case holds, but this time the probabilistic dependence is between the sentence of death and the skin's color of the victim (see figure 8.20, c)). The data for the model are:

$$n_0^v = \sum_{d=0}^1 \sum_{m=0}^1 n_{d0m} = 19 + 132 + 11 + 52 = 214 \quad \text{White victim}$$

$$n_1^v = \sum_{d=0}^1 \sum_{m=0}^1 n_{d1m} = 0 + 9 + 6 + 97 = 112 \quad \text{Black victim}$$

$$n_{10}^v = \sum_{m=0}^1 n_{10m} = 19 + 11 = 30 \quad \text{Sentenced to death, white victim}$$

$$n_{11}^v = \sum_{m=0}^1 n_{11m} = 0 + 6 = 6 \quad \text{Sentenced to death, black victim}$$

and we can jump directly to the evidence of the model:

$$p(n_{10}^v, n_{11}^v, n_0^v, n_1^v | \mathcal{H}_{10}) = \binom{n_0^v}{n_{10}^v} \frac{B(\alpha + n_{10}^v, \beta + n_0^v - n_{10}^v)}{B(\alpha, \beta)} \times \binom{n_1^v}{n_{11}^v} \frac{B(\alpha + n_{11}^v, \beta + n_1^v - n_{11}^v)}{B(\alpha, \beta)}$$

\mathcal{H}_{11}

Although the last case is the most complicated one, the resolution at this point is quite straightforward. From the PGM of figure 8.19, the death sentence is influenced by the skin's color of both victim and defendant, but these two occurrences are independent from each other. Hence, we have take into account both graphs a) and b) of figure 8.20, and factorize the various probabilities. Eventually we get the evidence


$$\begin{aligned} p(n_{10}^m, n_{10}^v, n_{11}^m, n_{11}^v, n_0^m, n_0^v, n_1^m, n_1^v | \mathcal{H}_{11}) &= \binom{n_0^m}{n_{10}^m} \frac{B(\alpha + n_{10}^m, \beta + n_0^m - n_{10}^m)}{B(\alpha, \beta)} \times \binom{n_1^m}{n_{11}^m} \frac{B(\alpha + n_{11}^m, \beta + n_1^m - n_{11}^m)}{B(\alpha, \beta)} \\ &\times \binom{n_0^v}{n_{10}^v} \frac{B(\alpha + n_{10}^v, \beta + n_0^v - n_{10}^v)}{B(\alpha, \beta)} \times \binom{n_1^v}{n_{11}^v} \frac{B(\alpha + n_{11}^v, \beta + n_1^v - n_{11}^v)}{B(\alpha, \beta)} \end{aligned}$$

Prior	Evidence			
	\mathcal{H}_{00}	\mathcal{H}_{01}	\mathcal{H}_{10}	\mathcal{H}_{11}
Uniform	97.5 %	1.2 %	1.3 %	≈ 0 %
Jeffrey	97.15 %	1.20 %	1.65 %	≈ 0 %

Table 8.1: Evidence computed for each model

Results

At this point we can evaluate the various evidences and see which is the best. We do it by considering both a uniform prior for the parameters $\alpha = \beta = 1$ and a Jeffrey's prior $\alpha = \beta = 1/2$. Table 8.1 shows the results of the exercise.

-  From the coding-computational point of view it's convenient to use the identity $p = e^{\log p}$ when we calculate probabilities. For example, evaluating the binomial factor when $n \gg 1$ is an heavy task, and using logarithms can really improve the accuracy of our results. So the best way to calculate the binomial factor is

$$\begin{aligned} \binom{n}{k} &= \frac{n!}{(n-k)!k!} = \exp[\ln(n!) - \ln(n-k)! - \ln(k!)] \\ &= \exp[\ln(\Gamma(n+1)) - \ln(\Gamma(n-k+1)) - \ln(\Gamma(k+1))] \end{aligned}$$



9. Communications channel

In the following section, we're going to depict a parallelism between information theory and data analysis. Lesson 15
 Information theory is pretty much about the study of *communications channel*, where data coming from a source get compressed, encoded, sent through a channel and then, once they get to their destination, they are decoded and decompressed to give the final output (see figure 9.1). The concern of information theory is to retrieve from the output as much information as possible about the input. 21/05 FC GC

We will show that *data analysis* can be seen, in a way, as a communication channel that tries to reproduce the data coming from the world by encoding only the important features in a generative model. However, while in information theory the input data can be whatever, in data analysis it matters a lot what kind of information we decide to retain in the compression phase of the communication channel. Here is where the concept of *relevance* comes into play.

To get started, let us justify this parallelism using an exercise taken from [Mac03].

Exercise — The data-processing theorem

The data processing theorem states that data processing can only destroy information. Prove this theorem by considering an ensemble (W, D, R) in which w is the **state** of the world, d is **data gathered**, and r is the **processed data**, so that these three variables form a Markov chain

$$w \rightarrow d \rightarrow r$$

that is, the probability $P(w, d, r)$ can be written as

$$P(w, d, r) = P(r|d)P(d|w)P(w). \quad (9.1)$$

Show that the average information that R conveys about W , $I(W, R)$, is less than or equal to the average information that D conveys about W , $I(D, W)$. This theorem is as much a caution about our *definition of information* as it is a caution about data processing!

Solution:

For any joint ensemble (X, Y, Z) the following chain rule for mutual information holds

$$I(X; Z) + I(X; Y|Z) = I(X; Y, Z) = I(X; Y) + I(X; Z|Y)$$

In this case w and r are independent given d so $I(W; R|D) = 0$; the previous equation becomes

$$I(W; R) + I(W; D|R) = I(W; D, R) = I(W; D) + I(W; R|D)$$

so finally we have

$$I(W;R) - I(W;D) = -I(W;D|R) \leq 0$$

This theorem shows that, within the thinking of information theory, we unavoidably destroy information when we process data. But how does this relate to the fact that when we analyze data we feel like gaining information? This is what the last sentence of the exercise is about: when we process data we are sifting them with respect to some choice we made about what is *relevant* in the data, and we get rid of all the other information we're not interested in. Information theory is a way of thinking about how to represent probabilities, not a tool that magically brings out what we need to know about the data.

9.1 Communications channels and information transmission

Communications channels are the way in which we can **transmit information**.

In order to describe a channel we can define some common elements. We start with a **source**, where the information is created, which sends data in a *random* way. This randomness can look weird, because when we communicate with someone we don't have the impression that the other person says letters at random. This notwithstanding, each letter of the alphabet has its own probability to be present in a word and in a sentence, so in a sense we're allowed to put a probability distribution $P(s)$ over the source. The data then go through a **compressor** that eliminates redundancies, and an **encoder** that produces the channel's input. The two concepts are separate: the compressor produces an output based on the structure of the source (for example it uses the information about the frequency of the letters in the alphabet), while the encoder wonders how the *noise* of the channel can be, and encodes the data in such a way that the receiver will be able to figure out all the content that was coming through the channel. We can therefore see how the compressor-decompressor part of the communication channel is concerned about the *structure of the source*, whereas the encoder-decoder part has to do with the *structure of the noise* acting on the channel.

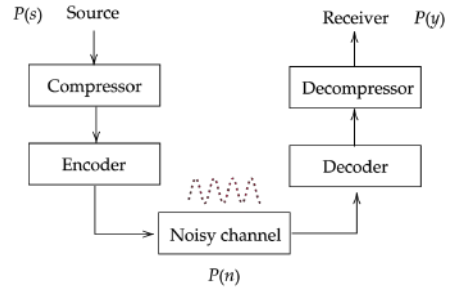


Figure 9.1: Communications channel

When we say structure, we always have in mind a probability distribution. So, in a communications channel, we have one probability distribution of the source, $P(s)$, one for the noise, $P(n)$, and one for the receiver, $P(y)$, and whole goal of communication is to infer $P(s)$ by observing $P(y)$. This interpretation allows to use all the tools of Bayesian inference introduced before, and in particular to design in the optimal way the structure of our communications channel.

Definition 9.1 — The optimal decoder. The *optimal decoder* for a channel code is the one that minimizes the probability of a block error. It decodes an output y as the input s that has maximum posterior probability

$$P(s|y) = \frac{P(y|s)P(s)}{\sum_{s'} P(y|s')P(s')}$$

9.1.1 Communication models

Communications channels can be described using models that allow to calculate how each information piece is transmitted, by associating to each element a probability to correctly transmit the data or to corrupt it during the process. By calling x the input and y the output, it is possible to define some channel examples.

■ **Example 9.1 — Binary symmetric channels.** Binary symmetric channels describe communication processes in which data can assume only the values 0 and 1. In an ideal, noiseless channel, every time the source sends 0 then the receiver gets 0, and the same with 1. If noise comes into play, however, it happens with probability f that the bit gets flipped, and therefore there are four conditional probabilities between x and y

- $p(y = 0|x = 0) = 1 - f$
- $p(y = 1|x = 0) = f$

- $p(y = 0|x = 1) = f$
- $p(y = 1|x = 1) = 1 - f$

In particular, the probability to corrupt the message is identical for each starting input value. ■

Using Bayesian tools it is also possible to estimate how much information the output conveys about the input through the mutual information

$$I(X;Y) = H(Y) - H(Y|X) = H(X) - H(X|Y) \quad \text{where} \quad H(Y|X) = H(f, 1-f)$$

if we have a binary channel then $H(f, 1-f) = H_2(f)$, which is called binary entropy. The amount of information shared between the two distributions allows to evaluate how noisy is the channel.

■ **Example 9.2 — Z channel.** The main difference between the binary symmetric channel and a Z channel is that the message "0" is always correctly transmitted, while "1" can be corrupted

- $p(y = 0|x = 0) = 1$
- $p(y = 1|x = 0) = 0$
- $p(y = 0|x = 1) = f$
- $p(y = 1|x = 1) = 1 - f$

■

9.2 Binary classifiers as Binary Asymmetric Channels

We can think binary classifiers, such as *Support Vector Machines*, *Random Forests* or *Boosting* as communications channels between the data and the labels that are assigned to them.

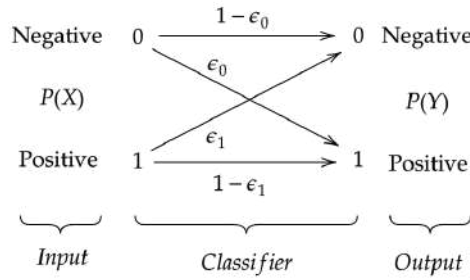


Figure 9.2: Binary classifier seen as a binary asymmetric channel

If X is the source and Y is the receiver, we can write the communications model as

$$P(X) = \begin{pmatrix} f \\ 1-f \end{pmatrix}, \quad P(Y) = \begin{pmatrix} (1-\epsilon_1)f + \epsilon_0(1-f) \\ \epsilon_1 f + (1-\epsilon_0)(1-f) \end{pmatrix}$$

$$P(Y|X) = \begin{pmatrix} 1-\epsilon_1 & \epsilon_0 \\ \epsilon_1 & 1-\epsilon_0 \end{pmatrix}, \quad P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)} = \begin{pmatrix} \frac{(1-\epsilon_1)f}{(1-\epsilon_1)f + \epsilon_0(1-f)} & \frac{\epsilon_1 f}{\epsilon_1 f + (1-\epsilon_0)(1-f)} \\ \frac{\epsilon_0(1-f)}{(1-\epsilon_1)f + \epsilon_0(1-f)} & \frac{(1-\epsilon_0)(1-f)}{\epsilon_1 f + (1-\epsilon_0)(1-f)} \end{pmatrix}$$

All we have to do in order to compute the posterior probability is to find some estimators for $\{\epsilon_0, \epsilon_1, f\}$. In the training phase of our ML model we have at disposal the true label for the input data, so we can compare them with the predictions of our model and compute the number of true positives TP, of true negative TN, and the number of false positives FP and false negatives FN. Using these quantities we can define the binary **confusion matrix** as

$$C_{bin} = \begin{pmatrix} TP & FP \\ FN & TN \end{pmatrix} \quad (9.2)$$

Using the entries of this matrix we can compute

$$\begin{cases} N_+ = TP + FN \\ N_- = FP + TN \\ N = N_+ + N_- \end{cases}$$

and infer the probabilities involved in the model as

$$\hat{P}(X) = \begin{pmatrix} \frac{N_+}{N} \\ \frac{N_-}{N} \end{pmatrix}, \quad \hat{P}(Y) = \begin{pmatrix} \frac{TP+FP}{N} \\ \frac{FN+TN}{N} \end{pmatrix}, \quad \hat{P}(Y|X) = \begin{pmatrix} \frac{TP}{N_+} & \frac{FP}{N_-} \\ \frac{FN}{N_+} & \frac{TN}{N_-} \end{pmatrix}$$

Finally, some plausible estimators for the parameters are

$$\begin{cases} \hat{\epsilon}_0 = \frac{FP}{N_-} = FPR \\ \hat{\epsilon}_1 = \frac{FN}{N_+} = FNR \\ \hat{f} = \frac{N_+}{N} \end{cases}$$

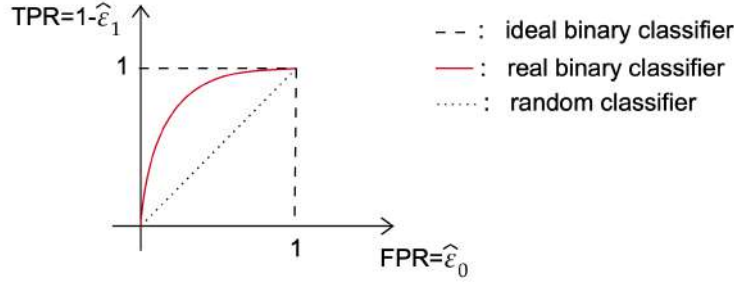


Figure 9.3: The Relative Operative Characteristic (ROC) curve for three different type of binary classifiers

From the definition of C many different accuracy metrics for binary classifiers have been defined, such as the sensitivity or true positive rate (TPR): $TPR = \frac{TP}{N_+} = \frac{TP}{TP+FN} = 1 - FNR$ or the specificity (TNR): $TNR = \frac{TN}{N_-} = \frac{TN}{TN+FP} = 1 - FPR$. Some of them can also be quite deceiving, such as accuracy $ACC = \frac{TP+TN}{N}$ since it does not take into account any eventual inter-class imbalance, that could lead to excellent accuracy if the dominant class is classified correctly also if the other one is completely mistaken. Given this arbitrariness on the choice of the metrics, it is way better to talk directly about the confusion matrix instead.

9.2.1 Evaluating Binary Classifiers

In the previous paragraph we saw how to write the posterior distribution of the input of a binary classifier given the output, provided some estimate of the biases obtained from the confusion matrix. Now we want to make precise statements about how to estimate $\hat{\epsilon}_0$ and $\hat{\epsilon}_1$ by means of Bayesian inference.

From a probabilistic point of view, an asymmetric binary classifier can be seen as a sequence of two tosses made with three coins $\{X, Y_+, Y_-\}$. We toss the first coin X , having a bias f , and we look at the result. If we get *head*, then we flip the coin Y_+ , which has a bias ϵ_0 . If instead we get a *tail*, then we flip the coin Y_- , that has a bias ϵ_1 .

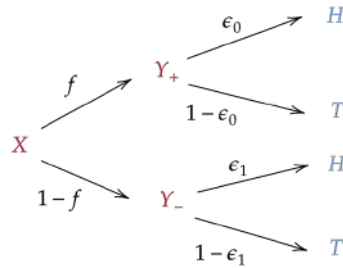


Figure 9.4: Coin tossing sequence as a binary classifier

The data we have are the entries of the confusion matrix. The Likelihood of the problem is the product of two binomial distributions

$$p(FP, FN | \epsilon_0, \epsilon_1, N_+, N) = \binom{N-N_+}{FP} \epsilon_0^{FP} (1-\epsilon_0)^{N-N_+-FP} \times \binom{N_+}{FN} \epsilon_1^{FN} (1-\epsilon_1)^{N_+-FN} \quad \text{Likelihood}$$

while, assuming ε_0 and ε_1 to be independent, we can choose for the parameters a *beta prior*

$$p(\varepsilon_0, \varepsilon_1 | \alpha, \beta, \delta, \gamma) = \text{Beta}(\varepsilon_0 | \alpha, \beta) \times \text{Beta}(\varepsilon_1 | \delta, \gamma) = \frac{\varepsilon_0^{\alpha-1} (1 - \varepsilon_0)^{\beta-1}}{B(\alpha, \beta)} \times \frac{\varepsilon_1^{\delta-1} (1 - \varepsilon_1)^{\gamma-1}}{B(\delta, \gamma)} \quad \text{Prior}$$

The posterior distribution can be easily determined by updating the parameters of the beta prior with the observations we made:

$$\begin{aligned} p(\varepsilon_0, \varepsilon_1 | FP, FN, N_+, N) &= \text{Beta}(\varepsilon_0 | \alpha + FP, \beta + N - N_+ - FP) \times \text{Beta}(\varepsilon_1 | \delta + FN, \gamma + N_+ - FN) \\ &= \frac{\varepsilon_0^{\alpha-1+FP} (1 - \varepsilon_0)^{\beta-1+N-N_+-FP}}{B(\alpha + FP, \beta + N - N_+ - FP)} \times \frac{\varepsilon_1^{\delta-1+FN} (1 - \varepsilon_1)^{\gamma-1+N_+-FN}}{B(\delta + FN, \gamma + N_+ - FN)} \quad \text{Posterior} \end{aligned}$$

The maximum of the posterior gives the following MAP estimators

$$\begin{aligned} \hat{\varepsilon}_0 &= \frac{FP + \alpha - 1}{N - N_+ + \alpha + \beta - 2} \\ \hat{\varepsilon}_1 &= \frac{FN + \delta - 1}{N_+ + \delta + \gamma - 2} \end{aligned}$$

and notice that in the case of a uniform prior ($\alpha = \beta = \delta = \gamma = 1$) we would get the MLE estimators we found above.

9.3 Relevance

As we mentioned above, the reliability of a communication channel can be quantified as the mutual information between the input X and the output Y , $I(X, Y)$. This quantity though, is also related to the probability distribution of the input, \mathcal{P}_X , so if we want to characterize our communications channel we should get rid of this dependence. This requirement leads straightforwardly to the concept of *capacity*.

Definition 9.2 — The capacity of a channel. The *capacity* of a communication channel Q is:

$$C(Q) = \max_{\mathcal{P}_X} I(X, Y)$$

In a way, this is like if we engineer the input probability distribution in such a way to convey as much information as possible by keeping the channel configuration fixed. The distributions (there can be more than one) \mathcal{P}_X that achieve the maximum $C(Q)$ are called *optimal input distributions* \mathcal{P}_X^* . These distributions \mathcal{P}_X^* for the source make X and Y to be the most interdependent as possible, and lead the encoding to compensate the noise of the channel at its best. In figure 9.5 we plot the probability of bit error p_b versus the rate of information conveyed by code R .

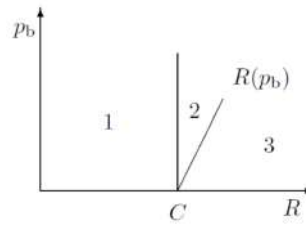


Figure 9.5: Portion of the R, p_b plane to be proved achievable (1, 2) and not achievable (3).

We can see that the capacity of the channel C is a sort of threshold for the rate R . For $R < C$ we can have a perfect reconstruction of the source (1), meaning that we can always find a \mathcal{P}_X^* that allows a perfect communication. The region (3) is the opposite, meaning that there is no possibility to reconstruct the signal, while (2) is something in between, namely we have limited possibilities to reconstruct the source.

Exercise What is the capacity of the binary symmetric channel for general f ?

Solution:

By symmetry, the optimal input distribution is $\{p_0 = 0.5; p_1 = 0.5\}$; then the capacity C is

$$C = I(X;Y) = H(Y) - H(Y|X) = H_2(0.5) - H_2(f) = 1 - H_2(f)$$

Without invoking symmetry, we can do this by computing the mutual information in the general case where the input ensemble is $\{p_0; p_1\}$:

$$C = I(X;Y) = H(Y) - H(Y|X) = H_2(p_0f + p_1(1-f)) - H_2(f)$$

that is indeed maximized for $p_0 = p_1 = 0.5$. ■

9.3.1 Relevance in information theory and data analysis

We have already described an existing parallelism between communications channel and the data analysis procedure. Data comes from the world, the source, and get compressed to retain only the relevant features. These information are then encoded in a model (e.g. the parameters of a Gaussian) capable to generate samples that are expected to resemble the data we collected. In this sense, there's a deep analogy between data analysis and pure information theory, as intended by information engineers. What is different in these two pictures is the **relevance**, i.e. what is deemed to be important to retain in the compression part.

For engineers that design a communications channel, what is relevant is *reconstruction*, namely the capability to recover the information being sent from the source to the receiver. This is a very neutral view: it doesn't matter what kind of data we have in input, the communications channel must be able to give an output that resembles as much as possible the original input. In the context of data analysis it is different.

Definition 9.3 — Relevance. *Relevance* is about detecting differences in observations y in the data space, \mathcal{D} , as required by a particular task.

This can be seen as the level of consistency between the data content and a specific task requirements. Then, relevance is not necessarily a property of the data itself, it is related to whether data was deemed to be valuable to be acquired as it was compatible with the user's interest. For example, we may not be interested in reconstructing exactly the input but, e.g. considering a classification task, what matters is just being able to assign a label from a certain set: this allows a massive compression of the original dataset.

Hence, the same dataset may be useful for different tasks by merely changing the relevance set \mathcal{L} , and the mapping $L: \mathcal{D} \mapsto \mathcal{L}$. Then, relevance is strictly bound to the distance metrics $d(y, y')$ in the data space, that depends on the level of compression we are performing on data.

In a nutshell: Again, never confuse the real world with our representation. The world is just out there, our representation instead is a map from the world respect to what we care about. This means that representations are made with a purpose, based on what we think is relevant. For example, if we want to study countries inside the EU, we should use a political map, while if we are hiking we should use a topography map. The world is the same but its representation has changed.

■ **Example 9.3 — Models and relevance: Electromagnetism.** The model and the relevance will depend on the task we want to undertake: for designing an antenna an engineer doesn't need to know QED! Maybe even the Maxwell equations will be too fundamental for this particular task, and it would be better to start from a good electrical engineering book. He will be using QED in an ontological way, but not in an epistemological one, because he doesn't need to know it. ■

■ **Example 9.4 — Choosing the relevance of triangles.** Consider a problem in which, given a two dimensional input image, we want to spot all the triangles and classify them as equilateral, isosceles, right or scalene. The setup is the following:

- **Computer vision part:** given the image y we have to build a *vertex detector* f^{-1} that maps the data space \mathcal{D} into the model space \mathcal{M} . Since each triangle has three vertices, to identify it 6 numbers are needed. This means that we have a 6-D model space: $\dim(\mathcal{M}) = 6$;
- **Classification part:** starting from a point in the model space, $\mathbf{x} \in \mathcal{M} = \mathbb{R}^6$, we want to assign it a label. Our relevance set \mathcal{L} , that is what we want to infer from data, is whether a triangle is equilateral,

isosceles, right or scalene, for a total of 4 labels: $|\mathcal{L}| = 4$.

For this kind of relevance set we do not actually care about rotations, position in the plane and scale: we need less information than knowing the 6 vertices coordinates. All that matters for this task is just two degrees of freedom: given the relation $\alpha + \beta + \gamma = \pi$, knowing two angles is all we need to undertake the desired classification task! The symmetries of the problem (rotational, translational and scale invariance) allow to shrink the 6-D model space into a 2-D model relevance space: $\dim(\mathcal{M}_r) = 2$. This corresponds to consider a sub-manifold of angles (α, β) , \mathcal{M}_r , in the space of models \mathcal{M} . The complementary sub-spaces form the 4-D invariance group: the Lie group, $\text{sim}(2)$, comprehending the translational, rotational and scale invariant degrees of freedom. ■

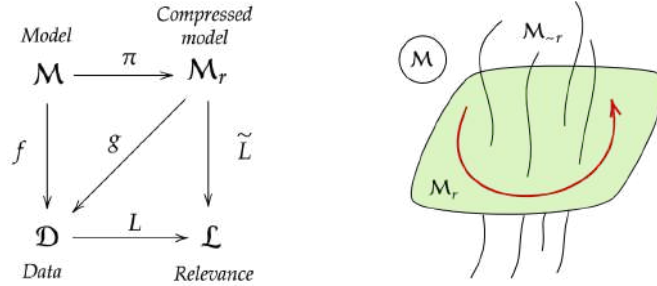


Figure 9.6: Left) Mapping from data space, model space, model relevance space and relevance space. Right) Sub-manifold \mathcal{M}_r of the model space \mathcal{M} .

The scheme in figure 9.6 (left) is a *commutative diagram* that represents the connections we discussed above. Here, the arrows connecting the various labels can either represent functions or probabilities. According to this scheme, we can write the mapping from data to labels $L: \mathcal{D} \rightarrow \mathcal{L}$ as

$$L \circ f = \tilde{L} \circ \pi \Rightarrow L = \tilde{L} \circ \pi \circ f^{-1}.$$

However, in real world things do not commute, meaning that f can be a *probabilistic* function, rather than deterministic. In this case we have to open up the commutative diagram introducing a function, Δ , that keeps track of the differences between data and the data sampled from the compressed model, as shown in figure 9.7.

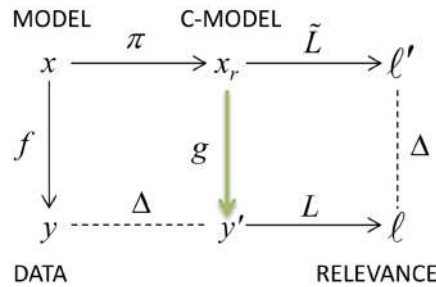


Figure 9.7: Real world non-commutative diagram.

We can also interpret this diagram in the frame of machine learning. In this case, we can consider the entire scheme and identify ℓ as the labels that we assign to the training set, and ℓ' as the labels inferred by the model. In this case thus, Δ will be the loss function of the ML model, for instance the categorical cross entropy.

But we could also look just at the leftmost part of the diagram, and interpret it as an unsupervised ML model. Now, if we think of a variational autoencoder, $\pi \circ f^{-1}$ will be the encoder, x_r the latent variables associated to the input data, namely a compressed representation that is supposed to retain only the relevant features of the data, and g will be the decoder. The generative model is therefore able to produce output data y' , that aim either to be similar to the original ones and at the same time to generalize them.

9.3.2 Information bottleneck

We can read figure 9.8 under the light of the triangles' example. Starting from the data space Y , which now can be thought as been made of sets of 6 coordinates vertices, we compress it in a generative model, S , that encompasses all the relevant features we care, namely the two angles. From this representation it's easy to put a label on each triangle representing whether it is equilateral, isosceles, right or scalene.

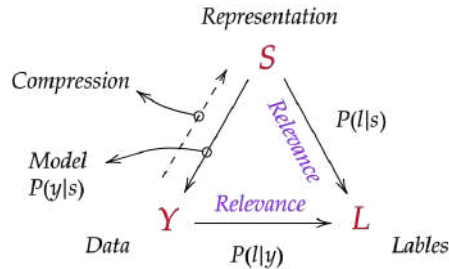


Figure 9.8: Relation between data, compressed representation and labels

In doing this we had to get rid of many redundant information which had no meaning for our task: we are far from the idea of integral reconstruction of the original data. To formally express this idea we can modify the optimization problem we saw for the communications channel in the following way

$$p^*(s|y) = \underset{p(s|y)}{\operatorname{argmax}} \underbrace{I(S, L)}_{\text{Relevance}} - \lambda \underbrace{I(Y, S)}_{\text{Compression}} \quad (9.3)$$

$= \mathcal{F}[p(s|y)]$

In practice, we have set up a communication channel between the data we have (Y) and what is relevant for us (L). This means that this is not the usual communication channel used in engineering, but rather a data analysis version of it. Thanks to Eq. (9.3) it is possible to find the best compromise between the *relevance* contribution $I(S, L)$, which strives to retain as much information as possible, and the *compression* term, $-I(Y, S)$, which instead would throw away everything. The λ parameter is a Lagrange multiplier that controls the trade-off between the two.

9.4 Machine learning

Lesson 16 What is the connection between physics and machine learning? In physics we put the emphasis on the *model*:
24/05 it has to capture the relevant information of the data and it must be able to make predictions in agreement
AM with observations.

LR In ML, instead, there is no model supposed to describe the data we observe: nobody talks about a model of cats or dogs when doing image classification. The focus is rather on the data and what is relevant in it. To catch what is important in the data it is useful to exploit some features (usually, vectors encoding important details). The connections between the two approaches can be established by identifying the model in physics with the features space in ML.

Here we assume everyone who gets to study those topics has already a background knowledge about what machine learning is, what is the biological inspiration of those techniques and which are the frameworks in which ML can be applied. This allows us to put more emphasis into the important relations that are established between these methods and the Bayesian inference introduced in the previous chapters. In particular, one may say that the unsupervised ML framework is the one corresponding to the "engineering information theory" communication, for which it is required to perform a reconstruction of the original data (i.e. cluster together words without previous knowledge of the message: $\mathbf{x} \mapsto \mathbf{x}'$) while the supervised ML framework focuses on relevance (map features \mathbf{x} s to a smaller, information-compressed space of labels \mathbf{c} s).

9.4.1 Statistics versus machine learning

While looking at the statistical modeling philosophies, two different paradigms arise. The first, which is more physics-based, is the "orthodox" point of view: the algorithms compose a "white box" (i.e., a computing

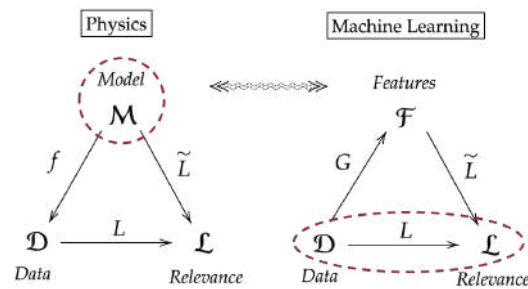


Figure 9.9: Schematic representation of the different approaches of physics and machine learning

structure which can be observed in detail and provides interpretability) which is able to link in a rigorous manner inputs and outputs. This extremely rational vision relies on critical decision making. On the other hand, instead of putting the emphasis on *how things are done* we could focus on *how good the result is*. This shifts the sight from the modelling unit to the outcomes and check methods. As a result, in the "machine learning" paradigm inputs and outputs are linked by a non-investigable black box which aims to highlight cool explanatory features and organized structures (such as decision trees) to allow the user to gain decision power on the quality of the results. This moves the attention from the abstract model to the observable performances, by using a methodological technique rather than the rigorous one of the other approach.

Hence, while trying to connect the concepts of models and ML one may keep in mind that the latter is data-driven and focuses on prediction, while the former relies on different assumptions and aims to explanation, i.e. to find the data themselves. Those are both the targets and also the reliability metrics for those approaches.

$$\begin{array}{cc} \text{Science} & \text{ML} \\ \text{Model} \rightarrow \text{Data} & \text{Data} \rightarrow \text{Features} \end{array}$$

However, it is of paramount importance to recall that the arrows above can be simply inverted by the means of Bayes' theorem: in fact, the theorem acts as an arrow-swapper in the form $p(\mathbf{y}|\mathbf{x}) = f(p(\mathbf{x}|\mathbf{y}))$ which allows to always find an inverse way to move from one space to the other, while exploiting probability. Hence, the concept of causality itself seems to lose clarity: one may start assuming the data are fixed and create a theorem or model *ad hoc* to describe them; on the other hand, if we know there is a particular theory behind a phenomenon we are more likely to recognize data according to it. This is a cyclic process with no real starting point nor end point, and the truth of what it describes has to be considered within the suitable boundaries.

One of the drawbacks of big ML architectures such as deep learning is the lack of interpretability: take as an example the filters of a convolutional neural network. They can be trained to recognize a particular label, but looking at their shape it is arguably impossible for the human brain to connect the filtered image to its theoretical interpretation. In general, we must recall that neural networks are not *biology-based* but only *biology-inspired*, and the working of NNs cannot fully reproduce the one of our brains. **We could say that human brains are an evolutionary solution to the statistical constraints of inference from experience.** So our algorithms don't work as our brain and indeed it's one of the most difficult concept to drop as soon as possible in order to develop good AI algorithms, because no one really knows as brain works. In the same fashion, birds are a solution to an engineering problem with physical constraints; the Wright brothers in developing the airplane understood the physical problem and then found an appropriate engineering solution different from birds even if they were **inspired** by the birds.

Consider the following example: we build a generative model starting from a neural network which is able to perform the traditional classification between two categories of images (e.g., rabbits and tigers). In particular, the network learns the existence of a set of mid-level features such as the geometry of an animal, its texture, the shape of its tooth, etc. With a prior knowledge given by the training over the true labels, the generative model could be able to produce new fake images, composing the subject in accordance to some internal probabilities. **As a result, we could see in the output pictures of creatures we have never seen before, yet our brain successfully manages to interpret and classify them.** In some way is like the generative model helps us to "dream" new subjects. This is a level of understatement that overcomes greatly the simple label distinction.

9.4.2 Labels

When working with supervised learning techniques we must be attentive to the way we set the labels. In general, we can organize labels in a "generative way" (like the one on the left in figure 9.10) or a "discriminative way" (on the right).

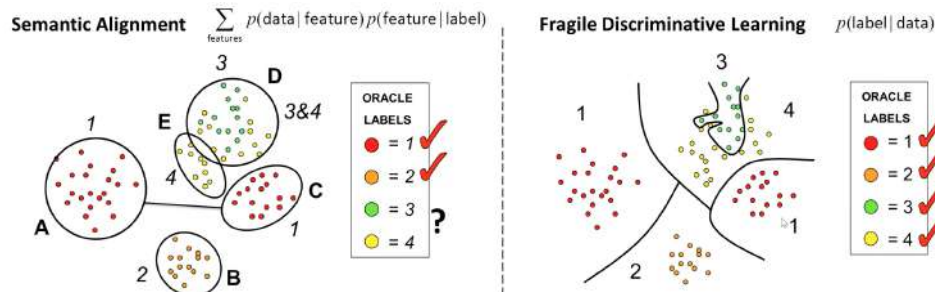


Figure 9.10: Labelling techniques

The former method is eventually more robust than the other, since it better characterizes the classes which the labels belong to. The discrimination model can be affected by biases determined by different experience, knowledge and background. As a result, while performing discrimination even human being could come up with different results.

But where do labels come from? Suitable labels must respect two fundamental characteristics: they must refer to something which can be detected (either via experimental apparatuses or by means of human senses) and must be useful to know about the world. Hence, before the actual ML implementation of supervised learning and usage of those labelled data, there is a whole processing of labelling information which is not independent of the context of creation. Moreover, while applying label we are trying to compress in just one word a whole set of features by trying to preserve the maximum amount of relevance. Since this results obviously in the waste of much information, labels are just a superficial transfer of intelligence to the ML algorithm.

What said above results in some great differences while trying to bring a ML framework back to a human-like behaviour. The most critical points are probably the fact that machine interpretations and abstract concepts are not commonly 100% compatible, which results in a lack of explanation and explainability. Moreover, the results of ML can be hard to transfer to the real world. This means that while using those techniques a certain care is required also in the "connection" between the input/output doors of the machinery and the real world outside.

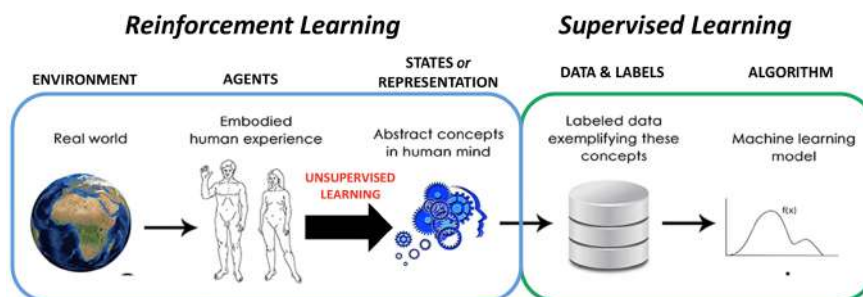


Figure 9.11: Scheme of ML from the origin to outcomes

We could eventually paraphrase what said above in terms of machine learning structures: as figure 9.11 shows, the labels generation and pre-processing can be attributed to a combination of reinforcement learning and unsupervised learning. In fact, the blue box could be modelled by the environment outside communicating with an agent inside the world which learns by his own actions in terms of the reward he receives by the outside after performing them. Hence, the human is schematized as a system with two devices: a sensorial, physical one which acquires data and a mental, computational one, which processes information.

This scheme is reported in figure 9.12: while focusing of the last, "mental" sector of the process, we notice it can be written as a logical unit with both an input a back-propagating output. It is quite straightforward to recognize the similarities with a ML classifier, which takes data in input as well but returns its answers in the form of labels. This point of view allows to simplify the model, since the whole reinforcement learning + supervised learning chain can be turned into a supervised learning + supervised learning one by simply understanding that the labelling procedure $Y \mapsto L$ that a machine learning algorithm has to perform is similar to the one $X \mapsto L$ performed by our brain. Thus, the role of ML is closer to understand which is the transformation $X \mapsto Y$ that models data, so to have the same labels (see Fig. 9.13).

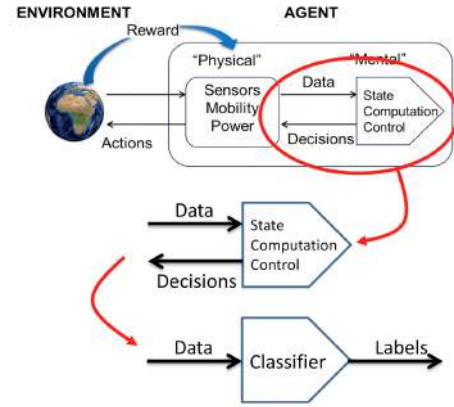


Figure 9.12: The ML point of view of human labelling

Even though this simplistic schema does not make justice to the complexity of human kind, it is still a more reliable model than the simple, deterministic "prepared-data-to-label" ML. In fact, the latter does not take into consideration the whole background of labels generation for training data, which can eventually be thought as a training session itself on a whole different database w.r.t. the one the new labels have to refer to, and this can lead to big trouble. Take this limits into account while evaluating the performances of your network.

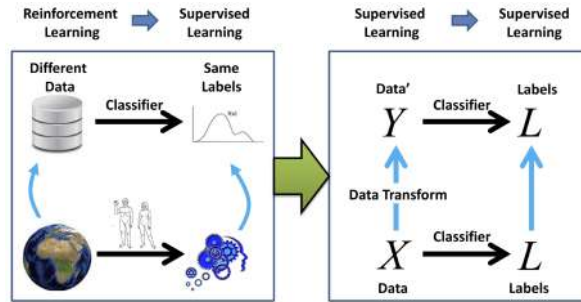


Figure 9.13: ML Label Interpretation.

9.4.3 The relationship of machine learning to Bayesian inference

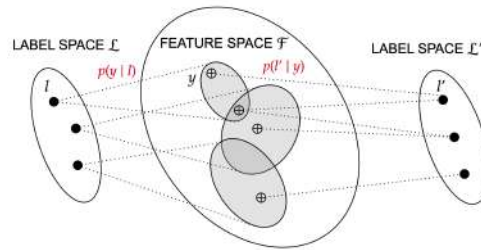


Figure 9.14:

We are now making a comparison between the supervised learning framework in Machine Learning and Bayesian Inference.

Now consider a classification problem: referring to Fig 9.14 , we have a label space \mathcal{L} and a feature space \mathcal{F} , that may or may not coincide with data space \mathcal{D} , depending on which information is relevant for the task

we are undertaking. Focusing on the ML training phase, each true label $l \in \mathcal{L}$ is represented by a vector $y \in \mathcal{F}$, which is a element of the training set. The training phase aims to assign to each y its own l' label in the assigned label space \mathcal{L}' . Errors in the classification can be generated in the regions of \mathcal{F} where there is an overlap of features representing different $l \in \mathcal{L}$, leading to the assignation of the wrong l' .

We can visualize this process more formally by the probabilistic graphical model in Fig. 9.15, in which we think supervised learning as a **deterministic optimization problem**.

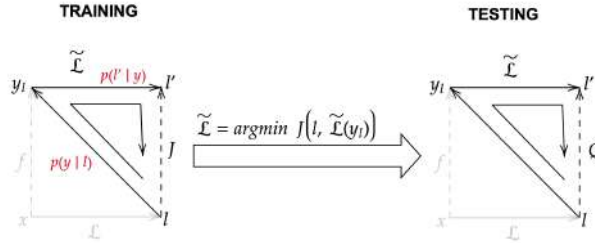


Figure 9.15: Probabilistic graphical model of supervised learning training and test procedures

The diagonal line represents $p(y|l)$, whereas the top line returns $p(l'|y)$. The dashed line encodes J , that is the cost function that compares l and l' . The whole loop represents the process of training and evaluating the results. Then the learning optimization problem is reframed as finding the optimal classifier

$$\tilde{\mathcal{L}} = \operatorname{argmin}_{\tilde{\mathcal{L}}} J(l, \tilde{\mathcal{L}}(y_l))$$

The cost function J is usually chosen with a view to mathematical convenience for the Stochastic Gradient Descent optimization process more than an effective description of the problem: a common choice is the *mean squared error*, i.e. the l_2 distance between the actual label l and the forecast one $l' = \tilde{\mathcal{L}}(y_l)$:

$$J = \sum_{i=1}^N |l_i - \tilde{\mathcal{L}}(y_i)|^2$$

This symmetric cost function is usually not fit for real-life problems: being 5 minutes early or 5 minutes late can lead to completely different results (e.g. if we have to take a train!), that are not taken into account by this J . After having obtained the optimal classifier we exploit it to predict the label l' on data we have not used till this moment: this test phase will be evaluated through the confusion matrix Q .

What the ML perspective hides is how these labels have been generated: we take for granted the process from which we obtain the true labelling l from the data $x \in \mathcal{D}$, but we actually don't have x but just its representation y with its own l associated.

That's why the Bayesian perspective is much more conscious, it takes into account in $p(l|y)$ this uncertainty in the labels associated to the y we got. So we can build the corresponding probabilistic diagram with the following equations

$$\begin{aligned} p(l|y) &= \frac{p(y|l)p(l)}{p(y)} && \text{Classifier} \\ p(y) &= \sum_{l \in \mathcal{L}} p(y|l)p(l) && \text{Evidence} \\ p(l'|l) &= \sum_{y \in \mathcal{Y}} p(l'|y)p(y|l) && \text{Confusion matrix} \end{aligned}$$

where $p(y|l)$ and $p(l)$ are both estimated from the training.

It is not possible to get rid of the uncertainty due to this overlap by changing the algorithm, it is intrinsic to data. Also, it has to be taken into account that there's no optimal *a priori* algorithm, but it depends on the considered task: this is expressed in the **No free-lunch theorem**, that express the concept of inductive bias, i.e. our algorithm will fail in some processes and succeed in others. In the Bayesian process, there is no training nor testing, that are substituted by conditional density estimation, and then no cost function J , since it is not needed: the Bayesian probability theory will lead straight to the confusion matrix $p(l'|l)$! In conclusion we can say that the Bayesian way of supervised-learning can be seen as a **conditional density estimation**.

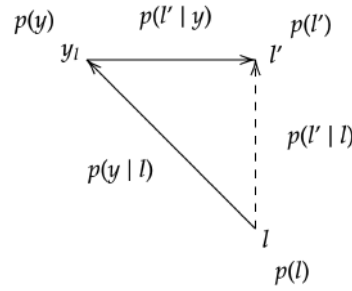


Figure 9.16: Conditional density estimation in Bayesian Inference

9.5 Predictive information

We will discuss all the concepts in the wake of the paper [BNT01]. The notion of predictive information (I_{pred}) Lesson 18 has a large importance in understanding how much we can learn about **future** from the **past**. Starting from 31/05 the assumption that non-predictive information is **useless**, a natural expression for predictive information is FC given by the **mutual information** between the past and the future themselves. TF

In order to understand better the meaning of this new quantity, we can introduce an example using the Ising model for a 1D spin lattice, associated to the Hamiltonian

$$H(\boldsymbol{\sigma}) = - \sum_{i,j} J_{ij} \sigma_i \sigma_j$$

where the matrix of interaction J is not restricted to the nearest neighbour interaction, but allows also long-range interactions. In particular, we are interested in studying the entropy of "words" built as a sequence of N consecutive spins as a function of N itself. Imagine to sample a random string of spins of length M according to the Boltzmann distribution $P(\boldsymbol{\sigma}) \propto \exp(-\beta H(\boldsymbol{\sigma}))$, with $\beta = 1$, and to fix the length of the world, $N < M$. Then, it will be possible to form $2^N - 1$ different words, and the entropy for this particular choice of N will be

$$S(N) = - \sum_{k=0}^{2^N-1} P_N(W_k) \log P_N(W_k) \quad (\text{bits}),$$

where $P_N(W_k) \approx n(W_k)/M$ is the frequency of the word W_k in the sequence.

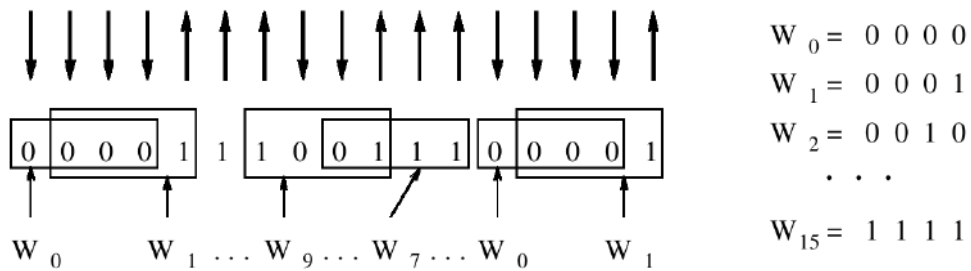


Figure 9.17: Calculating entropy of words of length 4 in a chain of 17 spins. For this chain $n(W_0) = n(W_1) = n(W_3) = n(W_7) = n(W_{12}) = n(W_{15}) = 2$, $n(W_8) = n(W_9) = 1$, and all other frequencies are zero. Thus, $S(4) \approx 2.95$ bits. Taken from [BNT01].

The entropy, as one could guess, increases with the size of the words, since larger sequences contains more spins and so more possible states are available. The key question is: what happen if we remove this trivial extensive behaviour of the entropy? Can we find something which is intrinsic in the problem, that in some sense is able to measure the complexity underneath the data? If we start looking at the **sub-extensive** part of the entropy, $S_1 \equiv S/N$, we can see how there is a deeper dependence on the "word" size associated to the interaction strength. In figure 9.18 we can see on the left the total entropy as a function of the word length, whereas on the right there is the same plot for S_1 . Three different couplings, J , between spins have been chosen, and for each of those we have a different scenario:

- If we have just one fixed parameter J , S_1 is flat. This means that making the chain longer is useless, we can learn J immediately.
- With short-range interactions, S_1 has a logarithmic growth, meaning that making the spin chains longer actually allows to learn more about the interaction.
- With long-range decaying interactions, we get a power law response meaning a significant increase of S_1 with N . This is the hint that there is something really complicated happening in the system.

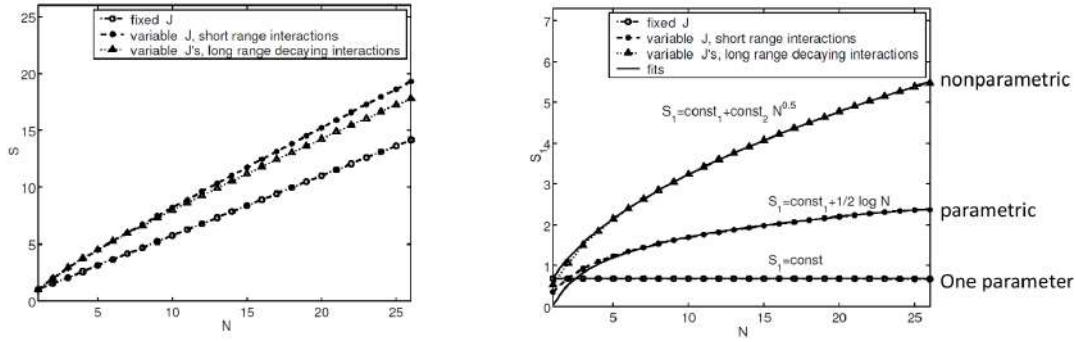


Figure 9.18: 1. Entropy as a function of the word length for spin chains with different interactions. The lines start from $S(N) = \log 2$, since the correlation length is much smaller than the chain length. 2. Subextensive part of the entropy as a function of the word length.

This kind of correlation structures can be found in many other cases, such as in text that can only be understood if we can link together words as we read it. This explains why we need a measure of complexity that can allow us to distinguish between random structures and **relevant** ones. Quoting the paper (p. 2411): *"An essential difficulty in quantifying complexity is to distinguish complexity from randomness."* In fact, as humans, we decide what is relevant merely on the base of the problem we're attempting to solve; randomness is then just what we deem to be negligible because we think it doesn't matter.

Typically, a very trivial measure of complexity can be achieved using **correlations**, for example studying the covariance matrix of the data if we can neglect moments of order higher than 2 and perform a Gaussian approximation. In a more general framework, mutual information is a better answer, since it allows to compare any probability without any kind of approximation (see "Correlation vs mutual information" paragraph). In ML, for instance, there are some measure of complexity, like **VC-dim**, but what we're exploring is something more fundamental based on information theory.

9.5.1 Mutual information between the Past and the Future

In this paragraph we will try to address the following questions: *What is the mutual information between the past and the future? If I know all the past, how much future can I predict?* Imagine that we observe a stream of data $x(t)$ over a time interval $-T < t < 0$; let all of these past data be denoted by the shorthand x_{past} . We are interested in saying something about the future, so we want to know about the data $x(t)$ that will be observed in the time interval $0 < t < T'$; let these future data be called x_{future} . In the absence of

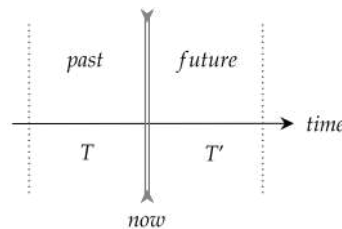


Figure 9.19: Past and future.

any other knowledge, futures are drawn from the probability distribution $P(x_{future})$, while observations of particular past data x_{past} tell us that futures will be drawn from the conditional distribution $P(x_{future}|x_{past})$.

The narrower shape of the conditional distribution can be compatible with the fact that it has smaller entropy than the prior distribution, and this entropy reduction is Shannon's definition of the information that the past provides about the future. We can write the average of this **predictive information** as

$$\mathcal{I}_{pred}(T, T') = \left\langle \log \left[\frac{P(\mathbf{x}_{future}|\mathbf{x}_{past})P(\mathbf{x}_{past})}{P(\mathbf{x}_{future})P(\mathbf{x}_{past})} \right] \right\rangle = \langle \log P(x_{future}, x_{past}) \rangle - \langle \log P(\mathbf{x}_{future}) \rangle - \langle \log P(x_{past}) \rangle$$

where $\langle \dots \rangle$ denotes an average over the joint distribution of the past and the future, $P(x_{future}, x_{past})$.

All this description can be mapped to the nomenclature involving entropies. In particular, the first term accounts for the joint entropy of past and future, the second for the entropy of the future, and the third for the past. Hence, we can write

$$\mathcal{I}_{pred}(T, T') = S(T) + S(T') - S(T + T')$$

that recalls the formula of the mutual information we are used to deal with. If there is no information conveyed between the T and T' windows, the predictive information is therefore zero.

We know two general facts about the behaviour of $S_1(T)$:

1. The corrections to extensive behavior are positive: $S_1(T) \geq 0$.
2. State that the entropy is extensive is equivalent to say

$$\lim_{T \rightarrow \infty} \frac{S(T)}{T} = S_0$$

But if the previous limit is true, then

$$\lim_{T \rightarrow \infty} \frac{S_1(T)}{T} = 0$$

Let's now take the following limit, where we **keep fixed** how much we go back in the past but we look for $T' \rightarrow \infty$ in the future

$$I_{pred}(T) = \lim_{T' \rightarrow \infty} \mathcal{I}_{pred}(T, T') = S_1(T)$$

This represents the best we can know about the future given the fixed window in the past T , namely the definition of **predictive information** in the paper.

If we have been observing a time series for a (long) time T , then the total amount of data we have collected is measured by the entropy $S(T)$, and at large T this is given approximately by $S_0 T$. But the predictive information that we have gathered **cannot grow linearly with time**, even if we are making predictions about a future which stretches out to infinity. There is only so much predictive information in the total information. As a result, of the total information we have taken in by observing x_{past} , only a vanishing fraction is of **relevance to the prediction**; this is what the **Law of diminishing returns** state.

$$\lim_{T \rightarrow \infty} \frac{\text{predictive information}}{\text{total information}} = \lim_{T \rightarrow \infty} \frac{I_{pred}(T)}{S(T)} \rightarrow 0$$

Looking at this limit it is straightforward to see that we can go back in the past collecting data as much as we want, but the predictive information becomes a smaller and smaller fraction of the total information. **Most of what we observe is irrelevant to the problem of predicting the future.** Furthermore, relevance is key for using information theory, just like quantifying ignorance via the prior is the key to Bayesian inference. Also notice that sorting a set of data does not lead to an increase in information, if done in a deterministic way: it will only make the search for useful insights in the data, i.e. what is relevant, way harder.

Consider the case where time is measured in discrete steps, so that we have seen N time points x_1, x_2, \dots, x_N . *How much have we learned about the underlying pattern in these data?* The more we know, the more effectively we can predict the next data point x_{N+1} and hence the fewer bits we will need to describe the deviation of this data point from our prediction: our accumulated knowledge about the time series is measured by the **degree to which we can compress the description of new observations**. On average, the length of the code word required to describe the point x_{N+1} , given that we have seen the previous N points, is given by:

$$\ell(N) = -\langle \log P(x_{N+1} | \underbrace{x_1, x_2, \dots, x_N}_{\text{observed time series}}) \rangle$$

measured in bits, where the average is done over everything, namely the joint distribution of all the $N + 1$ points. It is easy to see that

$$\ell(N) = S(N+1) - S(N) \sim \frac{\partial S(N)}{\partial N}$$

As we observe for longer times, we learn more and this word length decreases. Usually we define learning curves by measuring the frequency or costs of errors; here the cost is that our encoding of the point x_{N+1} is longer than it could be if we had perfect knowledge. This ideal encoding has a length that we can find by imagining that we observe the time series for an infinitely long time, $l_{ideal} = \lim_{N \rightarrow \infty} l(N)$, but this is just another way of defining the extensive component of the entropy S_0 . We can thus define a **learning curve** $\Lambda(N)$, that can be seen as an error rate measuring this improvement, defined as

$$\begin{aligned}\Lambda(N) &\equiv l(N) - \lim_{N \rightarrow \infty} l(N) = S(N+1) - S(N) - S_0 \\ &= S_1(N+1) - S_1(N) \approx \frac{\partial S_1(N)}{\partial N} = \frac{\partial I_{pred}(N)}{\partial N}\end{aligned}$$

This idea actually reflects the question: "how well we can predict the future, given the number of samples that we've seen in the past?".

The mathematical way to figure out what is going on inside the communication channel without leaving information theory run amok, but considering the relevance as part of the problem. We are not just doing the compression step but we need to preserve the relevance, otherwise the solution is very easy: simply forget everything. On the other hand compression is necessary because we don't have an unlimited amount of memory to store everything. Indeed this is the optimization problem we have already encountered:

$$p^*(s_{t+1}|a_t) = \underset{p(s_{t+1}|a_t)}{\operatorname{argmin}} \underbrace{\overbrace{I(A, S)}^{\text{Compression}} - \beta \overbrace{I(S, Y)}^{\text{Relevance}}}_{\mathcal{F}[p(s_{t+1}|a_t)]}$$

9.5.2 Determine I_{pred} for a Markov Process

The joint distribution of the N data points as a product is

$$P(x_1, x_2, \dots, x_N) = P(x_N|x_{N-1}, \dots, x_1) \dots P(x_2|x_1)P(x_1)$$

but due to the properties of a Markov process, we can write

$$P(x_n|\{x_{1 \leq i \leq n-1}\}) = P(x_n|x_{n-1})$$

hence the predictive information reduces to

$$I_{pred} = \left\langle \log \left[\frac{P(x_n|x_{n-1})}{P(x_n)} \right] \right\rangle$$

This seems to be great because we don't have to look many step behind in the past, but notice that in reality we've compressed the past in a potentially long state vector, with lot of dimensions and propagated through time. A useful example comes from Newtonian mechanics. If we have a discrete set of measurements x_t, x_{t+1}, \dots of a trajectory, we know that it's not enough to describe the dynamics, and we need also momenta. So instead to handle the full continuum trajectory $x(t)$ we introduce momenta that, given the position where we are, allow us to predict the future position. So in practice we don't need all the time steps behind, i.e. the full Markov Chain, but we need the full vector states of the Markov Chain.

We are left with two possibilities: or **we have a large memory**, namely we can remember all the steps x_t, x_{t+1}, \dots or **we have a system with a lot of states**. Consequently the states of a system is the memory encoded into the relevance.



10. Non parametric models

So far we've mainly talked about *parametric models*. Given our data, we select one or more models depending on parameters \mathbf{w} , and for each of those we use Bayesian inference to determine the posterior distribution of such parameters. Eventually, we compare all these models through their evidence and we retain the best one. The complexity of the model is thus represented by the set of parameters, no matter what the number of data samples is. Lesson 19 04/06 LR GC

Sometimes, though, the process underneath our data can be such complicated that it becomes difficult even to guess some model to describe it. In these cases, non parametric models allow to fit and generalize our observations without any particular assumptions on the process that originated them. Intuitively, if we have a large number of data we could feel justified in increasing the number of parameters more and more, and eventually send it to infinity. These are indeed data-driven methods conceptually more similar to machine learning rather than physics. One can argue that using a huge number of parameters (or an infinite number!) can lead to the problem of overfitting, and indeed generalization is a key concern of ML. However, formulate our problem in terms of Bayesian inference allow us to handle generalization issues from the beginning, because the prior we assign to the model is a natural regularizer.

The road map for this chapter is the following: we firstly introduce the Gaussian process and how to use it to make predictions, then we'll spend some time on the intuition behind the Gaussian process and the concept of kernel and Green's functions; eventually, we introduce a more advanced application of this concept, namely how to generalize the Gaussian process from the Euclidean space to manifolds. In the last part of the chapter we will also introduce the idea behind the Dirichlet process, an important tool that will allow us to talk about probability over probability distributions.

10.1 Gaussian process

10.1.1 Nonlinear regression: parametric approach

Suppose we are given a set of data $\{\mathbf{x}^{(n)}, y_n\}_{n=1}^N$, where $\mathbf{X}_N \equiv \{\mathbf{x}^{(n)}\}_{n=1}^N$ is the set of D dimensional input vectors and $\mathbf{y}_N \equiv \{y_n\}_{n=1}^N$ is the corresponding set of target values, which we'll assume to be real values in the following discussion. Our goal is to infer a nonlinear function $f(\mathbf{x})$ parametrized by parameters $\mathbf{w} = \{w_k\}_{k=1}^K$ that is assumed to describe the data \mathbf{X}_N , i.e.

$$\mathbf{y}_N = f(\mathbf{X}_N) + \boldsymbol{\eta} \quad (10.1)$$

where $\boldsymbol{\eta}$ is the noise; in words, \mathbf{y}_N are noisy versions of some function acting on \mathbf{x} .

The inference of $f(\mathbf{x})$ is described by the posterior probability distribution

$$P(f(\mathbf{x})|\mathbf{y}_N, \mathbf{X}_N) = \frac{P(\mathbf{y}_N|f(\mathbf{x}), \mathbf{X}_N)P(f(\mathbf{x}))}{P(\mathbf{y}_N|\mathbf{X}_N)} \quad (10.2)$$

where the first term on the right-hand side, $P(\mathbf{y}_N|f(\mathbf{x}), \mathbf{X}_N)$, is the probability of the target values given the function $f(\mathbf{x})$, which in the case of regression problems is often assumed a separable Gaussian distribution; and the second term, $P(f(\mathbf{x}))$, is the prior distribution of functions assumed by the model.

Once we choose a set of basis functions $\{\phi_k(\mathbf{x})\}_{k=1}^K$, we can think to decompose f as a linear combination of these functions

$$f(\mathbf{x}; \mathbf{w}) = \sum_{k=1}^K w_k \phi_k(\mathbf{x}). \quad (10.3)$$

We can choose whatever orthogonal basis, such as Hermite or Laguerre polynomials, spherical harmonics and so on. For example, if we take radial basis functions centered at fixed points $\{\mathbf{c}_k\}_{k=1}^K$,

$$\phi_k(\mathbf{x}) = \exp \left[-\frac{(\mathbf{x} - \mathbf{c}_k)^2}{2r^2} \right] \quad (10.4)$$

then $f(\mathbf{x}, \mathbf{w})$ is a nonlinear function of the input \mathbf{x} .

In a nutshell: Here, once the basis is fixed the complexity of the function f is somehow measured by the number of parameters, K , that we put into our model. In principle, one has to be careful in increasing the number of parameters, because a too powerful model is very likely to overfit the data. In general, one can set an upper bound on the generalization error

$$\text{Generalization error} \leq \sqrt{\frac{\text{Complexity measure}}{N}} \quad (10.5)$$

and hence the assumption of non parametric models is that the complexity grows sub-linearly with the number of data (e.g. N^β , $\beta < 1$), giving a finite generalization error. This is the reason why we can think to play with an infinite representation of our model without incurring in overfitting problems.

Now, since f is a deterministic function of \mathbf{w} , then the inference problem is actually on the parameters. The corresponding posterior reads

$$P(\mathbf{w}|\mathbf{y}_N, \mathbf{X}_N) = \frac{P(\mathbf{y}_N|\mathbf{w}, \mathbf{X}_N)P(\mathbf{w})}{P(\mathbf{y}_N|\mathbf{X}_N)} \quad (10.6)$$

Having obtained the posterior for \mathbf{w} , predictions are then made by marginalizing over the parameters:

$$P(\mathbf{y}_{N+1}|\mathbf{y}_N, \mathbf{X}_{N+1}) = \int d^D \mathbf{w} P(\mathbf{y}_{N+1}|\mathbf{w}, \mathbf{x}^{(N+1)})P(\mathbf{w}|\mathbf{y}_N, \mathbf{X}_N) \quad (10.7)$$

10.1.2 From parametric models to Gaussian processes

Let us consider the same setup we used before: N input data in a D dimensional space, \mathbf{X}_N , the corresponding observed outputs \mathbf{y}_N and a set of K basis functions ϕ_k . We then introduce the $N \times K$ matrix obtained by evaluating each basis function on each of the input vectors

$$R_{nk} \equiv \phi_k(\mathbf{x}^{(n)}) \quad (10.8)$$

and also the vector \mathbf{f}_N as the vector of values of $f(\mathbf{x})$ at the N points,

$$f_n \equiv \sum_k R_{nk} w_k. \quad (10.9)$$

If the prior distribution of \mathbf{w} is a multivariate normal with zero mean,

$$P(\mathbf{w}) = \mathcal{N}(\mathbf{w}; \mathbf{0}, \sigma_w^2 \mathbf{1}),$$

then \mathbf{f} , being a linear function of \mathbf{w} , is also Gaussian distributed with mean zero. The covariance matrix of \mathbf{f} is

$$\mathbf{G} = \langle \mathbf{f} \mathbf{f}^T \rangle = \langle \mathbf{R} \mathbf{w} \mathbf{w}^T \mathbf{R}^T \rangle = \mathbf{R} \langle \mathbf{w} \mathbf{w}^T \rangle \mathbf{R}^T = \sigma_w^2 \mathbf{R} \mathbf{R}^T, \quad (10.10)$$

so the prior distribution of \mathbf{f} is

$$P(\mathbf{f}) = \mathcal{N}(\mathbf{f}; \mathbf{0}, \mathbf{G}) = \mathcal{N}(\mathbf{f}; \mathbf{0}, \sigma_w^2 \mathbf{R} \mathbf{R}^T). \quad (10.11)$$

The matrix \mathbf{G} is called **Gram's matrix**, and we will return on it later on. Eq. (10.11) is the defining property of what is called **Gaussian process**.

Definition 10.1 — Gaussian process. The probability distribution of a function $f(\mathbf{x})$ is a Gaussian process if, for any finite selection of points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}$, the density $P(f(\mathbf{x}^{(1)}), f(\mathbf{x}^{(2)}), \dots, f(\mathbf{x}^{(N)}))$ is a Gaussian.

What about the target values? The \mathbf{y}_N vector we measure doesn't have infinite precision, and we therefore assume it to differ by additive Gaussian noise of variance σ_v^2 (measurement variance) from the corresponding function value f_N :

$$y_n = f(\mathbf{x}^{(n)}; \mathbf{w}) + \sigma_v \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, 1) \quad (10.12)$$

Then \mathbf{y} also has a Gaussian prior distribution,

$$P(\mathbf{y}) = \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{C})$$

where we introduced the covariance matrix

$$\mathbf{C} = \mathbf{G} + \sigma_v^2 \mathbf{1} = \sigma_w^2 \mathbf{R} \mathbf{R}^T + \sigma_v^2 \mathbf{1}.$$

In particular, the (i, j) entry of \mathbf{C} is

$$C_{ij} = \sigma_w^2 \sum_k \phi_k(\mathbf{x}^{(i)}) \phi_k(\mathbf{x}^{(j)}) + \delta_{ij} \sigma_v^2.$$

Notice that this is a $N \times N$ matrix, meaning that increasing the number of basis functions we use to define f doesn't affect the dimensionality of the problem.



Pay attention to not be deceived by the name ‘‘Gaussian process’’. We're not assuming that our input data are normally distributed; the Gaussian uncertainty is on the y -axis, namely on the fitting function f and on the output data y .

■ **Example 10.1** Let us consider the one-dimensional case with the radial basis function defined in (10.4). Then, instead of considering a finite number of basis functions, we see what happens if we take the limit $K \rightarrow \infty$. To do that, we multiply and divide the above equation for ΔK , being the distance between two consecutive means of the basis functions in the x -axis, and we assume that $\sigma_w^2 / \Delta K \rightarrow \text{const.}$ as $K \rightarrow \infty$. This is reasonable, since as the number of weights increase they also get closer, and consequently their uncertainty must become smaller. This way the sum above becomes an integral, and the covariance matrix can be computed as

$$\begin{aligned} C_{ij} &\propto \int dk \phi_k(x^{(i)}) \phi_k(x^{(j)}) + \delta_{ij} \sigma_v^2 \\ &= \int dk \exp\left[-\frac{(x^{(i)} - k)^2}{2r^2}\right] \exp\left[-\frac{(x^{(j)} - k)^2}{2r^2}\right] + \delta_{ij} \sigma_v^2 \\ &= \sqrt{\pi r^2} \exp\left[-\frac{(x^{(i)} - x^{(j)})^2}{4r^2}\right] + \delta_{ij} \sigma_v^2 \end{aligned}$$

■

The example above shows a quite remarkable result. Instead of specifying the prior distribution on functions in terms of basis functions and priors on parameters, the limit $K \rightarrow \infty$ allows to summarize the prior of the output data through a covariance matrix \mathbf{C} given by

$$C_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) + \sigma_v^2 \delta_{ij} \quad (10.13)$$

where k is a sort of covariance function called **kernel**. Evaluating the kernel on the data gives what we anticipated to be the **Gram's matrix**

$$G_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}).$$

The covariance matrix of the prior is therefore given by the Gram's matrix plus a diagonal matrix that quantifies the uncertainty on the data. This additional term turns out to be actually quite important, since prevents the covariance to be singular and hence allows to invert it. Notice that at this stage there's no need anymore to talk about a *set of basis functions*: once we choose the kernel function the problem is fully specified.

In conclusion, the prior probability of the N target values \mathbf{y} in the data set is:

$$P(\mathbf{y}) = \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{C}) = \frac{1}{Z} e^{-\frac{1}{2} \mathbf{y}^T \mathbf{C}^{-1} \mathbf{y}} \quad (10.14)$$

10.1.3 Using a given Gaussian process model in regression

Now that we've seen how the prior of a Gaussian process looks like, we wonder how can we make predictions from it. In other words, we want to infer the value of the data y_{N+1} given that we observed \mathbf{y}_N .¹ The joint density $P(y_{N+1}, \mathbf{y}_N)$ is a Gaussian, so the conditional distribution

$$P(y_{N+1}|\mathbf{y}_N) = \frac{P(y_{N+1}, \mathbf{y}_N)}{P(\mathbf{y}_N)} \quad (10.15)$$

is also a Gaussian. If we indicate as \mathbf{C}_{N+1} the covariance matrix of the vector $\mathbf{y}_{N+1} \equiv (y_1, \dots, y_{N+1})^T$, then we can write it as

$$\mathbf{C}_{N+1} = \begin{bmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^T & \kappa \end{bmatrix}$$

The posterior distribution (10.15) is given by

$$P(y_{N+1}|\mathbf{y}_N) \propto \exp \left[-\frac{1}{2} [\mathbf{y}_N \ y_{N+1}] \mathbf{C}_{N+1}^{-1} \begin{bmatrix} \mathbf{y}_N \\ y_{N+1} \end{bmatrix} \right]. \quad (10.16)$$

Instead of inverting \mathbf{C}_{N+1} , we can write the inverse in terms of \mathbf{C}_N and \mathbf{C}_N^{-1} as

$$\mathbf{C}_{N+1}^{-1} = \begin{bmatrix} \mathbf{M} & \mathbf{m} \\ \mathbf{m}^T & m \end{bmatrix}$$

where

$$\begin{aligned} m &= (\kappa - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k})^{-1} \\ \mathbf{m} &= -m \mathbf{C}_N^{-1} \mathbf{k} \\ \mathbf{M} &= \mathbf{C}_N^{-1} + \frac{1}{m} \mathbf{m} \mathbf{m}^T. \end{aligned}$$

When we substitute these matrices into Eq. (10.16) we find

$$P(y_{N+1}|\mathbf{y}_N) = \frac{1}{Z} \exp \left[-\frac{(y_{N+1} - \hat{y}_{N+1})^2}{2\hat{\sigma}_{N+1}^2} \right] \quad (10.17)$$

where

$$\hat{y}_{N+1} = \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{y}_N = \mathbf{k}^T (\mathbf{G}_N + \sigma_v^2 \mathbf{1})^{-1} \mathbf{y}_N \quad (10.18)$$

$$\hat{\sigma}_{N+1}^2 = \kappa - \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{k} = \sigma_f^2 - \mathbf{k}^T (\mathbf{G}_N + \sigma_v^2 \mathbf{1})^{-1} \mathbf{k}. \quad (10.19)$$

These last two quantities are what we were looking for: they describe the expected value of the prediction y_{N+1} and the uncertainty associated. The expression for the variance has actually a nice interpretation. κ is not just the kernel function evaluated in the point we want to make the prediction, but it's also the corresponding diagonal entry of the covariance matrix. This means that κ represents the variance that the prior assigns to the predicted value before having seen the data. The formula above is telling us that when we stick the data into the problem, then we are shrinking the spread of possible values that our predictions can take. This idea is sketched in figure 10.1.

Here below we report a simple implementation in R of a Gaussian process with an RBF kernel. The data consists on ten samples following a sinusoidal signal to which we added some Gaussian noise, and we assigned to each sample an uncertainty equal to the width we set for the noise. In this case we don't just make a single prediction, but many at one time.

¹for the sake of simplicity we restrict the discussion to just one prediction, but it is straightforward to extend the method for many predicted points at one time.

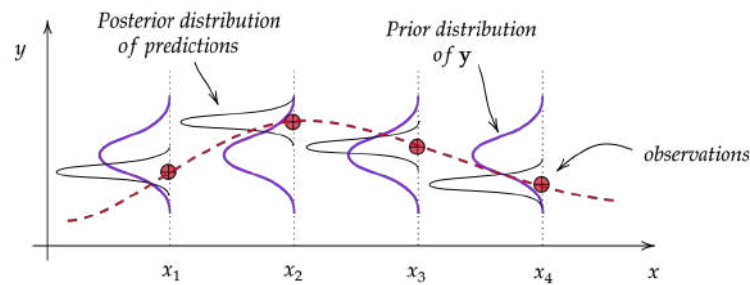


Figure 10.1: In the Gaussian process we assign a Gaussian uncertainty to the output values, y , and we update it with the knowledge coming from the data

```
library(matlib)
library(tidyverse)
library(plot.matrix)
library(mvtnorm)

# Generating data set
n_obs <- 10
x.obs <- seq(0, 5, length.out = n_obs)
y.obs <- sin(x.obs) + rnorm(n_obs, mean = 0, sd = 0.3)
y.sigma <- 0.3 # Uncertainty assumed on the measurements y.obs
obs_df <- tibble(x=x.obs, y=y.obs)

# Defining RBF kernel
rbf <- function(x1, x2, r, A){
  A*exp(-0.5 * (x1 - x2)^2/r^2)
}

# Prior distribution
C_N <- outer(x.obs, x.obs, rbf, 1, 1) + diag(y.sigma, n_obs, n_obs)
C_N.inv <- inv(C_N)
y.prior <- rmvnorm(5, sigma=C_N.inv)
cols <- c("dodgerblue", "firebrick1", "darkorchid1", "chartreuse3", "gold")
plot(x.obs, y.prior[1,], type="l", lwd=3, col=cols[1],
     main="Samples from prior distribution",
     xlab="x", ylab="y", ylim=c(-4,4))
for (i in 2:5){
  lines(x.obs, y.prior[i,], lwd=3, col=cols[i])
}

# Posterior distribution
n_pred <- 50 # Number of predictions
x.pred <- seq(-1, 7, length.out = n_pred)
k <- outer(x.obs, x.pred, rbf, 1, 1)
k.T <- outer(x.pred, x.obs, rbf, 1, 1)
kappa <- outer(x.pred, x.pred, rbf, 1, 1)

mu.pred <- k.T %*% C_N.inv %*% y.obs # Mean of predictions
cov.pred <- kappa - (k.T %*% C_N.inv %*% k) # Covariance matrix of predictions
sigma <- sapply(diag(cov.pred), sqrt) # Standard deviation of predictions
pred_df <- tibble(x = x.pred, y = mu.pred)

# Plotting results
ggplot(pred_df, aes(x=x, y=y)) +
  geom_ribbon(aes(x=x, ymin=(y - 2*sigma), ymax=(y + 2*sigma), fill="95%CL"),
            alpha=0.5) +
  geom_ribbon(aes(x=x, ymin=(y - sigma), ymax=(y + sigma), fill="68%CL"),
            alpha=0.5) +
  geom_line(aes(x=x, y=y, colour="Mean"), lwd=2) +
  geom_point(data = obs_df, aes(x=x.obs, y=y.obs, colour="Observations"),
            size=3) +
  geom_errorbar(data = obs_df, aes(ymin = (y.obs - y.sigma),
                                   ymax = (y.obs + y.sigma)), col="red", width=0.1) +
  scale_fill_manual(name="", values=c("95%CL" = "cornflowerblue",
```

```

"68%CL" = "dodgerblue")) +
scale_colour_manual(name="", values=c("Mean" = "darkblue",
                                       "Observations" = "red")) +
ggtitle("RBM Gaussian process predictions") +
theme_minimal()

```

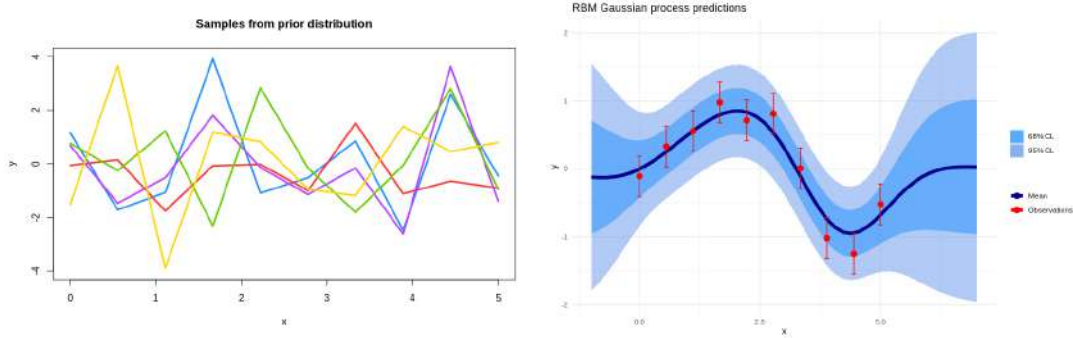


Figure 10.2: Results of the Gaussian process from the script above

10.2 The intuition behind Gaussian process and kernels

In the previous section, we saw that what characterizes a Gaussian process is the kernel. As we are going to see later on, the choice of the correct kernel is crucial for advanced tasks such as clustering in a manifold, so it's worth building some intuition on it. In general, we can interpret the kernel as an *infinite matrix* operating on some Hilbert space of functions, \mathcal{H} . Most of the properties of the kernel have indeed their matrices counterpart:

Property	Kernels	Matrices
Hermitian (symmetric)	$k^*(x, x') = k(x', x)$	$\mathbf{A}^T = \mathbf{A}$
Positive – definite	$\int f^*(x)k(x, x')g(x') dx dx' \geq 0 \forall f, g \in \mathcal{H}$	$\mathbf{x}^T \mathbf{A} \mathbf{y} \geq 0$
Eigenfunctions	$\int k(x, x')\phi_i(x')dx' = \lambda_i\phi_i(x)$	$\mathbf{A}\mathbf{v}_i = \lambda_i\mathbf{v}_i$
Mercer representation	$k(x, x') = \sum_{i=1}^{\infty} \lambda_i\phi_i(x)\phi_i^*(x')$	$\mathbf{A} = \sum_{i=1}^N \lambda_i\mathbf{v}_i\mathbf{v}_i^T$

Kernels are widely used in many applications, such as Support Vector Machines (SVM) in machine learning. Here, kernels are used to map data into an infinite dimensional Hilbert space, where points become linearly separable (kernel trick) and one can perform the classification task, eventually returning into the original data space.

10.2.1 Gram's matrix and covariance matrix

We saw before that evaluating the kernel on the input data gives the Gram's matrix, which is almost the covariance matrix of the prior distribution (Eq. 10.14). Although the knowledge of the kernel function is sufficient to accomplish the non-parametric interpolation, for the sake of intuition it can be useful to express the Gram's matrix in terms of the basis functions. Let's go back for a moment to the case in which we have K basis functions $\Phi = \{\phi_k\}_{k=1}^K$. Then, the set of basis functions induces a non-linear map from the D dimensional data space to a K dimensional Hilbert space, \mathcal{H} , given by

$$\Phi: \mathbb{R}^D \ni \mathbf{x}^{(i)} \mapsto \mathbf{r}^{(i)} = \left(\phi_1(\mathbf{x}^{(i)}), \dots, \phi_K(\mathbf{x}^{(i)}) \right)^T \in \mathcal{H}.$$

The entries of the Gram's matrix are then obtained as the dot product of the transformed samples in the feature space

$$G_{ij}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \langle \Phi(\mathbf{x}^{(i)}), \Phi(\mathbf{x}^{(j)}) \rangle = [\mathbf{R}\mathbf{R}^T]_{ij}, \quad \text{where } R_{ik} \equiv \phi_k(\mathbf{x}^{(i)})$$

In the expression above, \mathbf{R} is a $N \times K$ matrix, and overall \mathbf{G} is a $N \times N$ matrix, where recall that N is the number of data samples. A similar relation occurs when we want to compute the sample covariance matrix of our transformed data set:

$$\text{Cov}(\mathbf{R}) = \frac{1}{N} \sum_{i=1}^N (\mathbf{r}^{(i)} - \boldsymbol{\mu}_r)(\mathbf{r}^{(i)} - \boldsymbol{\mu}_r)^T = \frac{\mathbf{R}_c^T \mathbf{R}_c}{N}$$

where \mathbf{R}_c indicates the centered data set; notice that in this way the covariance is a $K \times K$ matrix. This means that, when we take the Gaussian process limit $K \rightarrow \infty$, the Gram's matrix keeps the same shape, whereas the covariance matrix becomes infinite! Interestingly, though, it can be shown that the two matrices share the same positive eigenvalues. The $K \rightarrow \infty$ limit therefore acts to the covariance matrix by adding a huge null space.

- S** From the computational point of view, the $N \times N$ structure of the Gram's matrix can be a weakness of the Gaussian process. In fact, the number of operations needed to invert such a matrix scales as $O(N^3)$ (better performances can be achieved if the matrix is *sparse*), making the algorithm quite computationally demanding for large numbers of data points.

10.2.2 Kernel as local average

The fancy computer scientists' way of seeing a Gaussian process is to picture each data point in a graph and link them together with edges weighted by the kernel's value between each two nodes (see figure 10.3). We therefore say that the kernel imposes a geometry over the data points.

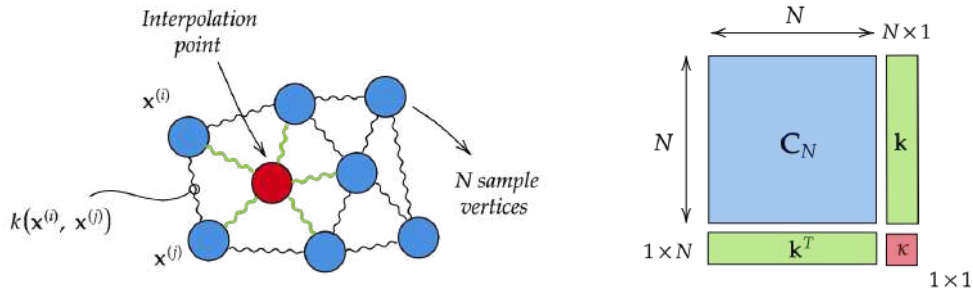


Figure 10.3: The kernel induces a graph on data

Computers scientist would say that the kernel induces a graph on data, that is to say that *the kernel finds a reasonable way to connect the new data for averaging*. Indeed, the interpretation that we can give to Eq. (10.18) - (10.19) is to estimate the function f at the interpolation point x_{N+1} (y_{N+1} in the notation above) using a linear combination of the observations $y(\mathbf{x})$ to form a weighted local average, where the weights are given by the kernel. Figure 10.3 could help visualize how the graph (left) corresponds to the matrix \mathbf{C}_{N+1} used before (right).

10.3 Diffusion on a manifold

We haven't talked about the choice of the kernel yet. How to get the proper kernel for a specific Gaussian process? Let us consider again the diffusion process without any external potential. The diffusion equation in $\mathcal{H} = \mathbb{R}^d$ is

$$\partial_t p = D \nabla^2 p$$

which is solved by the Green's function

$$G(\mathbf{x}, \mathbf{x}'; \tau) = \frac{1}{(4\pi D\tau)^{\frac{d}{2}}} \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{4D\tau}\right).$$

This is actually identical (up to a normalization factor) to the radial basis function kernel we saw before

$$k(\mathbf{x}, \mathbf{x}'; r) = \exp \left[-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2r^2} \right]$$

This is not merely a coincidence. In a very informal way we can say that the Laplacian operator has a sort of local averaging property (originating the *mean value property* of the harmonic functions, which are solutions of the Laplace equation $\nabla^2 f = 0$). As we saw before, this is also the kind of behaviour we would like a kernel to exhibit, so we can make the educated guess that **the kernel function is given by the solution of the diffusion equation in a certain Hilbert space \mathcal{H}** .

10.3.1 How to infer a generic kernel

We can generalize this point of view for data belonging to a manifold embedded in an higher dimension input space, considering $y = f(\mathbf{x})$, $\mathbf{x} \in \mathcal{M}$. In fact, finding a kernel for a Gaussian process is just a fancy way of performing a local average in \mathcal{M} using as weights its geodesic distance. The Green's function can be obtained by solving the diffusion equation on \mathcal{M} , and it will give us the kernel function for the Gaussian process.

If \mathcal{M} has a metric tensor g , denoting

$$\Delta_g = \frac{1}{\sqrt{|g|}} \partial_i (\sqrt{|g|} g^{ij} \partial_j f)$$

the Laplace-Beltrami operator, then the Green's function $G(\mathbf{x}, \mathbf{x}'; \tau)$ will be retrieved solving the following diffusion equation on a manifold

$$(\partial_\tau - \Delta_g) G(\mathbf{x}, \mathbf{x}'; \tau) = \delta(\mathbf{x} - \mathbf{x}')$$

having set the initial condition

$$G(\mathbf{x}, \mathbf{x}'; \tau = 0) = \delta(\mathbf{x} - \mathbf{x}') \quad (10.20)$$

From this $G(\mathbf{x}, \mathbf{x}'; \tau)$ we can easily define the kernel as

$$k(\mathbf{x}, \mathbf{x}' | \sigma_f^2, \tau) = \sigma_f^2 \frac{G(\mathbf{x}, \mathbf{x}'; \tau)}{\sqrt{G(\mathbf{x}, \mathbf{x}'; \tau) \cdot G(\mathbf{x}, \mathbf{x}'; \tau)}} \stackrel{*}{=} \sigma_f^2 \frac{G(\mathbf{x} - \mathbf{x}'; \tau)}{G(0; \tau)} \quad (* \text{isotropic, homogeneous case})$$

where, as usual, σ_f^2 is the variance of the prior distribution. In this way we don't have to guess the kernel: we can properly derive it just knowing the structure of the input space \mathcal{M} .

■ **Example 10.2 — Data on a spherical surface S^2 .** Let's consider for example a dataset that belongs to the spherical surface S^2 embedded in \mathbb{R}^3 . The Laplace-Beltrami operator reduces to the Laplacian in spherical coordinates

$$\Delta_{S^2} = \frac{1}{\sin \theta} \frac{\partial}{\partial \theta} \left(\sin \theta \frac{\partial}{\partial \theta} \right) + \frac{1}{\sin^2 \theta} \frac{\partial^2}{\partial \phi^2}$$

The diffusion equation can be solved isolating the spatial and time dependencies, where the spatial equation reduces to the eigenvalues problem for Δ_{S^2} :

$$\Delta_{S^2} P_l(\cos \gamma) = l(l+1) P_l(\cos \gamma)$$

in which we denote with $P_l(\cos \gamma)$ the Legendre polynomial of degree l and

$$\cos \gamma = \cos(\phi - \phi') \cdot \sin \theta \sin \theta' + \cos \theta \cos \theta'$$

The resulting Green's function is

$$G_t(\theta, \phi; \theta', \phi') = \frac{1}{4\pi} \sum_{l=0}^{\infty} (2l+1) \cdot \exp^{-l(l+1)\tau} P_l(\cos \gamma)$$

from which the *Legendre kernel* can be obtained

$$k(\theta, \phi; \theta', \phi' | \sigma_f^2, t) = \sigma_f^2 \frac{\sum_{l=0}^{\infty} (2l+1) \cdot \exp^{-l(l+1)\tau} P_l(\cos \gamma)}{\sum_{l=0}^{\infty} (2l+1) \exp^{-l(l+1)\tau}}$$

This set of basis functions can be interpreted as the RBF kernel bent on the surface of the sphere: the RBF-generated kernel constitutes a local average in an Euclidean space, whereas the Legendre kernel implements local averaging with respect to the geodesics on the sphere. The spatial bandwidth of prior τ is the only free parameter left and it regulates the spread of the average range: this determines the complexity of the function that we can represent with this kernel. This fact can be visualized in Fig. 10.4: if τ is sufficiently small we can represent high-complexity functions (possibly leading to overfitting), whereas when τ increases we average over wider and wider regions, until reaching a constant function over the sphere surface.

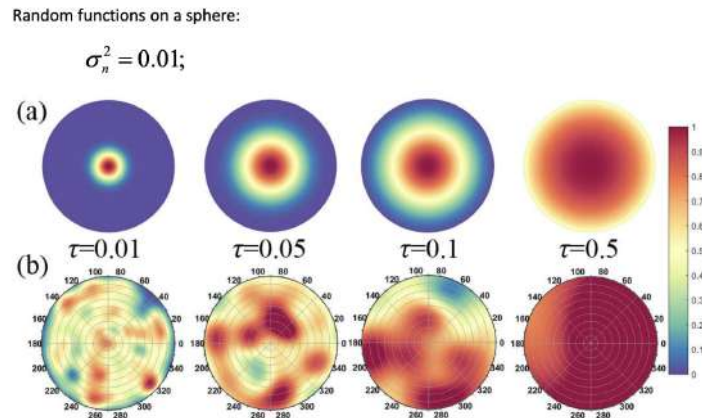


Figure 10.4: (a) The angular correlation function as a function of geodesic distance γ on the unit sphere (b) Examples of functions on the sphere randomly drawn from the Gaussian process for various values of τ [DB18]

In a nutshell: We briefly summarize the entire workflow of GPR (Gaussian process regression). There could be situations in which the model that describes our data is too complex and we are not able to describe it in terms of few fundamental parameters. In those cases, we can use a ML "inspired" way to proceed, where the number of parameters could be in principle arbitrary large, under the assumption that the complexity grows sub-linearly with the number of parameters. By the fact that we want to perform a non-linear regression with a function that we actually don't know, we can instead consider its expansion in terms of some basis, like the RBF one. Since it turns out that the Gram's matrix is all we need, instead of specifying the basis we can equivalently make the choice on the kernel. However, it's not clear yet which kernel we should use in our problem, so up to now, we have just shifted the problem from guessing the non-linear regression function to the guess of the kernel. Then, we have seen that the solution of the usual diffusion equation, i.e. the associated Green's function, has the same form as the RBF kernel. This suggests that the input space \mathcal{M} , where our data naturally live and which can be described by a metric tensor g , induces an opportune diffusion equation on the manifold \mathcal{M} through the Laplace-Beltrami operator Δ_g . Finally, Green's solution of the corresponding diffusion equation on the manifold can be used to calculate the kernel that we should use. The only thing left to the user in the algorithm is τ , which determines the complexity of the function that we want to represent with this kernel.

10.4 Dirichlet process

Given a bunch of data, we can estimate their probability density function by arranging them into bins of a histogram. But how sure are we about the solution we found? After all the bin size is an arbitrary choice we made that influences pretty much the result. There is an underlying problem here about estimating the "probability of probabilities".

Lesson 20
07/06
AM
AZ

10.4.1 The Dirichlet distribution

Let us consider an example: we have inferred a discrete pdf in the form $P = \{p_i\}_i$ where $i = 1, \dots, K$ and thus we are able to calculate the entropy $H = -\sum_i p_i \log(p_i)$. However, the values $\{p_i\}_i \equiv \mathbf{p}$ have not been

directly measured: we have an estimate of their values according to the data we observed (for example by the maximum likelihood estimator, i.e. the number of counts per bin in a spectrum divided by total counts). Let us consider the case in which $K = 3$: out of 20 observations we get bin outcomes (7, 3, 10), which leads to estimate probabilities $\mathbf{p} = (7/20, 3/20, 1/2)$ with $\sum_i p_i = 1$. From here, we could estimate the entropy.

It is absolutely fair to assume that if we run the experiment again, the outcomes in the bins will be different and so will be the entropy (but the amount of information the environment gave us is the same). Hence, we want to be able to figure out from the data the probabilities of the probabilities \mathbf{p} (and consequently of H), i.e. perform Bayesian inference of the pdf. In order to do so, it will be helpful to exploit the function below:

Definition 10.2 — Dirichlet distribution and simplex. The Dirichlet distribution of order K (where $K \geq 2$) is a parametric distribution of params $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_K)$, $\alpha_i \geq 0 \forall i$ which has form:

$$\text{Dir}(\mathbf{p} | \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^K p_i^{\alpha_i - 1} \quad (10.21)$$

where $B(\boldsymbol{\alpha})$ is the multivariate beta function of the so called **concentration** parameters $\boldsymbol{\alpha}$. Of course, the values p_i must be suitable candidates for probabilities, i.e. they must respect:

$$\sum_{i=1}^K p_i = 1 \quad \text{and} \quad p_i \geq 0 \quad \forall i \in \{1, \dots, K\} \quad (10.22)$$

The requirements above define the so-called "simplex", which represents the simplest possible $K - 1$ -dimensional polytope in any given space. Familiar examples are the 2-simplex (triangle) and the 3-simplex (tetrahedron). In higher dimensions, the simplex is a generalization of those concepts. One may also notice that its definition is the same of an hyper-sphere with the use of the L1 norm instead of the L2 one.

Therefore, the simplex constitutes the domain of the Dirichlet distribution, which is a parametric function over it. Let us consider again the previous example: for $K = 3$ the $K - 1$ simplex is a triangle, whose three sides are the axes of p_1, p_2 and p_3 , respectively. Each internal point of the triangle represents a tuple (p_1, p_2, p_3) by opportune projection over the axes. Figure 10.5 reports the Dirichlet distribution for the case $K = 3$ for different $\boldsymbol{\alpha}$ parameters: we can notice that if those are all the same (upper plots) the distribution is symmetric for \mathbf{p} and the most probable values are $p_i = 1/3 \forall i$. We can also observe that the bigger α_i are, the more concentrated the distribution will be. While having different α_i , finally, the distribution narrows over different regions of the simplex. **The goal of Bayesian inference is to update $\boldsymbol{\alpha}$ according to the data.**

Why is this distribution so important? Suppose that we are given K boxes (or bins), each one having associated a probability p_k . Then, we have N objects that are arranged inside those boxes according to the probability vector $\mathbf{p} = \{p_k\}_{k=1}^K$. The probability distribution of the vector of number of occurrences $\mathbf{n} = \{n_k\}_{k=1}^K$ is given by multinomial distribution (i.e. we are dealing with K mutually exclusive possible outcomes which occur with probabilities \mathbf{p}):

$$\text{Mult}(\mathbf{n} | \mathbf{p}) = N! \prod_{k=1}^K \frac{p_k^{n_k}}{n_k!} \quad (10.23)$$

It can be proven that **the Dirichlet distribution is the conjugate prior of a multinomial likelihood**. We can therefore treat the probabilities \mathbf{p} as the parameters of a multinomial distribution and then use Bayesian inference to get the posterior distribution associated. Thus, for the problem of estimating the probability of bins it is sufficient to set a Dirichlet prior, perform the experiment and update the $\boldsymbol{\alpha}$ concentration parameters exploiting the update rule:

$$\alpha_i \rightarrow \alpha'_i = \alpha_i + n_i \implies \boldsymbol{\alpha} \rightarrow \boldsymbol{\alpha}' = \boldsymbol{\alpha} + \mathbf{n} \quad (10.24)$$

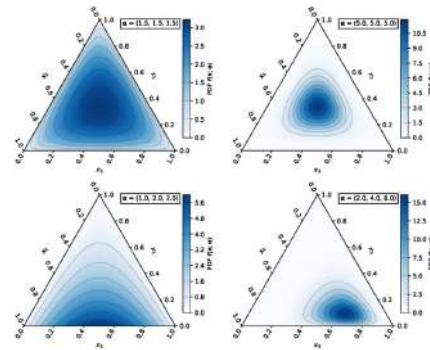


Figure 10.5: Example of Dirichlet distribution over the simplex

where n_i is the number of counts for the i -th bin. This allows to decrease dramatically the computational time of the evidence (integration over the simplex).

Now, in the previous paragraphs we learned how to increase our knowledge about the probabilities \mathbf{p} by exploiting our data so to have in the end a whole pdf for \mathbf{p} from which extract the statistical quantities of our interest. However, it becomes crucial the choice of the parameters $\boldsymbol{\alpha}$ for the prior, since we do not want it to dominate the likelihood (otherwise we would not learn from the experiment outcomes). It is common practice to adopt the so-called "symmetric Dirichlet distribution" as prior, for which at the very beginning $\alpha_k = \tilde{\alpha} \forall k$ (as a consequence all probabilities are initially equal). There are several possible choices of $\tilde{\alpha}$:

- $\tilde{\alpha} = 0$: maximum likelihood estimator;
- $\tilde{\alpha} = 1$: Laplace's rule;
- $\tilde{\alpha} = 1/2$: Jeffreys' prior;
- $\tilde{\alpha} = K^{-1}$: Schurmann-Grassberger estimator;

The value of $\tilde{\alpha}$ simply sets how peaked the prior will be around the central point of the simplex.

So our bayesian inference of a probability distribution becomes

$$Dir(\mathbf{p}|\boldsymbol{\alpha}') = \frac{Mult(\mathbf{n}|\mathbf{p})Dir(\mathbf{p}|\boldsymbol{\alpha})}{DirMult(\mathbf{n}|\boldsymbol{\alpha})}$$

and the predictive posterior is

$$DirMult(\mathbf{n}'|\boldsymbol{\alpha}') = \int_{\text{K-simplex}} d\mathbf{p} Mult(\mathbf{n}'|\mathbf{p})Dir(\mathbf{p}|\boldsymbol{\alpha}')$$

10.4.2 Paper discussion about PDFs of probabilities

The paper [IB02] deals with further problems in estimating quantities while dealing with uncertainty on probabilities. In particular, it emphasizes the enormous impact that the choice of the prior has on the final results. Generally speaking, we start from a prior assumption on \mathbf{p} (the Dirichlet prior) and a set of observations \mathbf{n} to produce our posterior $P(\mathbf{p})$ (I). Then, we perform other operations on the posterior to get the quantities of interest (e.g., the entropy S) (II). To put into perspective:

$$\mathbf{n} \xrightarrow{\text{I}} P(\mathbf{p}) \xrightarrow{\text{II}} S$$

It looks like there is no problem in this: we express our ignorance about the probabilities \mathbf{p} in the prior, get the posterior and compute the entropy. But looking at the whole process (I+II) we notice that we are getting the entropy *by starting from a prior which does not express our ignorance about the entropy itself*, and this could lead to some wrong conclusions.

Let us give an example: we are given a ball of size R in D -dimensions and we know that it contains a set of N sample points, \mathbf{X} . We could face two different problems:

1. If the problem we face is "estimate the probability of finding a sample at location \mathbf{x} ", to express maximum level of ignorance in the prior we choose $p_1(\mathbf{x}) = 1/V(R)$.
2. On the other hand, if the problem is instead "estimate the probability of finding a sample at distance $r \leq R$ from the origin", we select a different prior, i.e. $p_2(r) = 1/R$.

Both p_1 and p_2 are uniform distributions, but in different domains and with different assumptions. However, it is known that because of the so-called "phase space effect", the volume of an hyper-sphere of dimension $D \gg 1$ concentrates itself on the surface. As a result, if we express our ignorance as for the first problem and set p_1 as the prior, this will dramatically change our point of view about the second problem: in fact, for high dimensions if we assume that the samples are uniformly distributed in the volume, then we know they will be not uniformly distributed on the radial dimension, but there will be a much higher probability of finding them at $r \simeq R$. So making assumptions on something which is not our final target could really shift the focus and misinterpret our ignorance a lot.

Let us move back to the processes I and II above: the paper reports a very interesting example in which an encoding exercise is studied where we are dealing with $K = 1000$ bins. In this scenario, it can be shown that making innocent, ignorant assumption on the distribution of probabilities (by fixing a value of $\tilde{\alpha}$ in a symmetric Dirichlet prior) actually leads to a prior for the entropy S which is all but uniform! On the contrary, it is indeed very peaked and very dependent on the value of $\tilde{\alpha}$ we have chosen, at the point that setting a value of $\tilde{\alpha}$ leads to setting the value of S almost uniquely, regardless of how many samples \mathbf{n} we have for the inference. Of course, this is a huge disaster, since we are not exploiting our data at all to enlarge

our knowledge on the posterior. These examples are huge warnings about using quantities we have inferred for further calculations (it is not the same to infer a posterior pdf or a statistical quantity such as mutual information or entropy).

If our final target is the estimation of the entropy S (or the mutual information) of N samples in K bins, as in the example above, then the issues we faced can be solved by using the **Nemenman-Shafee-Bialek (NSB)** approach: since, as we saw, fixing a value of $\tilde{\alpha}$ leads to a strong condition on the final S estimation, their idea is to define a better prior distribution for $\tilde{\alpha}$ which makes use of the Jacobean of the expectation value of entropy $\langle S \rangle_{\mathbf{n}=0, \tilde{\alpha}} \equiv \mu_0(\tilde{\alpha})$ to have a prior for the entropy which is approximately uniform and equal to $1/\log(K)$.

Then, by performing a change of variables the probability of S can be written as the integral of a narrow function over μ_0 which is computation friendly and leads to an estimations of S given the observations \mathbf{n} that is not strongly dependent on the choice of $\tilde{\alpha}$ anymore.

10.4.3 The Dirichlet process

Lesson 21 The Dirichlet process is based on the idea that, given a binning problem, either the number and structure of
11/06 bins and how to partition data are unknown. How does data determine the “binning” structure? It will be
GC given by both the types of bins and the data assignments: the combination of this two factors will constitute our model. The Dirichlet process represents a non-parametric Bayesian approach to make data assignment.

In order to have a more straightforward explanation we start describing the **Chinese restaurant process (CRP)** example, which is nothing but a simplified version of the Dirichlet process where some variables are marginalized out, recasting the problem to a clustering one. We have data that are coming in, and we have to divide them in partitions; then the analogy will be the following:

Data \mathbf{x} \longrightarrow Customers
Clusters \longrightarrow Tables
Parameters θ_k \longrightarrow Dishes

In each table there are many different dishes, so that the customers on that table can try what they prefer from that selection. The choice of a customer to sit at a given table will be related both on the dishes on that table and on the number of people already sat there. If we consider a random assignment of old or new bin, the conditional probabilities of sitting at table k at which at time t are already sat $n_k(t)$ people on a total of N_t or to sit at a new table are the following:

$$p(\text{table } k | n_k(t)) = \frac{n_k}{N_t - 1 + \alpha}$$

$$p(\text{new table} | n_k(t)) = \frac{\alpha}{N_t - 1 + \alpha}$$

in which $\alpha \in [0, +\infty)$ is the concentration parameter of the symmetric Dirichlet distribution. α expresses the probability that if the customer shows up, a new table is created or, in our case, express the probability that a new data create a completely new bin (that is quite anomalous!). In order to understand what is the role of α in the process we can consider the two extreme cases:

- if $\alpha = 0$, none new tables are created, so our representation will never grow. This corresponds to a massive compression in which data are all the same according to the model and all is classified in just one bin;
- if $\alpha \rightarrow \infty$, then the probability to go in any previous existing table is zero: each customer has his own table and we are not performing any compression, i.e. we will create more and more bins.

Intermediate values of α regulate the two opposite compression behaviours, and from here we understand that α in a way controls the complexity of the model.

Notice that data can grow unconditionally, and for this reason we talk about “non-parametric model”: our representation is not finite before we see the data, it grows with the data without any limit. Our histogram is here treated as a clustering model: before clustering there is, for sure, more information, but the data processing effect is to retain only the relevant part of it.

Why should customers join one or the other table? In the formulation above we just considered a random assignment of old or new bin. We can now build a more detailed model that takes into consideration the

likelihood of the table choice

$$p(\text{table } k | n_k(t)) = \frac{n_k \cdot F(\mathbf{x} | \theta_k)}{N_t - 1 + \alpha}$$

$$p(\text{new table} | n_k(t)) = \frac{\alpha \cdot G(\theta | \mathbf{x})}{N_t - 1 + \alpha}$$

Here, $F(\mathbf{x} | \theta_k)$ is the likelihood function of the customers \mathbf{x} in the case in which the parameters of the model (dishes) are already set. If the bin/table is new we have a completely different approach: $G(\theta | \mathbf{x})$ expresses the prior probability of inferring the model parameter θ by the piece of data \mathbf{x} . Notice that the bins' structure is determined by the choice of $F(\mathbf{x} | \theta_k)$ and $G(\theta | \mathbf{x})$.

In many clustering algorithms, such as K-means, we need to specify the number of clusters beforehand: this means we should know a-priori the complexity of the process that produced the data! In the Dirichlet processes, instead, the number of bins is inferred and not set.

Here, we compromise between a non-parametric approach, where we allow the number of bins to grow to infinity, ($K \rightarrow \infty$), and a parametric approach, where instead we fix the complexity of the model to a certain constant value ($K = C$).

■ **Example 10.3 — Gaussian mixture model.** An example can be made from the Gaussian Mixture Model that we've seen in the chapter about information geometry. Here, we attempt to recognize clusters on a manifold, where each cluster (i.e. the multidimensional counterpart of a bin) is represented by a multivariate Gaussian. For K clusters we can write

$$p(\mathbf{x}) = \sum_{k=1}^K \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \cdot \pi_k(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

The Dirichlet process is the non-parametric approach for determine the binning structure. Instead of fixing a-priori some finite K , we include in the problem a probability density $p(K | \mathbf{x})$.

Recalling Fig. 8.16, we therefore have

$$p(\mathbf{x}) = \int F(\mathbf{x} | \boldsymbol{\theta}) \pi(\boldsymbol{\theta}) d\boldsymbol{\theta} \quad (10.25)$$

where choosing $F(\mathbf{x} | \theta_k)$ as the parametric likelihood makes $G(\theta | \mathbf{x})$ to be a kind of posterior for a single data point; in addition we couple it to a non-parametric combinatorial engine, represented by the prior $\pi(\theta)$, that can be given by the Dirichlet process or its marginalized version, the chinese restaurant process. ■

Statistical manifold

We can study this hybrid parametric-non parametric model with an information geometry approach:

- **we partition the data we have and form local models** $\pi_0(\theta)$: this encoding phase goes from the data space to the manifold of local models \mathcal{M} that we know, because we've selected our parametric likelihood function $F(\mathbf{x} | \theta_k)$, that induces \mathcal{M} .
- Once we have π_0 we can perform operations on it, like filtering and smoothing and Fourier analysis.
- After applying whatever operation/regularization we like, **we reach a global representation** $\hat{\pi}(\theta)$ and from this set of local models we decode the density estimate $p(\mathbf{x}) = \int d\mathbf{w} p(\mathbf{x} | \theta) \pi(\theta)$ in the data space
- Finally we can perform simulations and generate synthetic data, compatible with the structure of our dataset.

In a nutshell: We have a parametric set of models but we don't know how complicated they are in their space; however, we know the manifold and we are learning π_0 over the manifold, with the hope to achieve the global representation $\hat{\pi}(\theta)$.

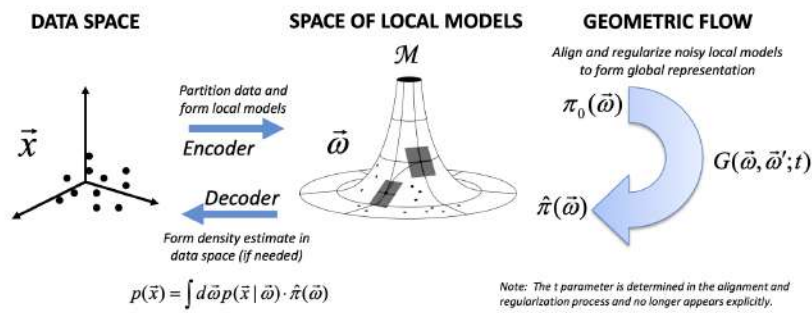


Figure 10.6: Information geometry approach to the parametric/non parametric composition approach
Beware the notation: $\mathbf{w} \longrightarrow \boldsymbol{\theta}$



Bibliography

- [AKS14] C. Albert, H. R. Künsch, and A. Scheidegger. “A simulated annealing approach to approximate Bayes computations”. In: *Statistics and Computing* 25.6 (Sept. 2014), pp. 1217–1232. ISSN: 1573-1375. DOI: 10.1007/s11222-014-9507-8. URL: <http://dx.doi.org/10.1007/s11222-014-9507-8> (cit. on pp. 39, 40).
- [AUS16] C. Albert, S. Ulzega, and R. Stoop. “Boosting Bayesian parameter inference of nonlinear stochastic differential equation models by Hamiltonian scale separation”. In: *Phys. Rev. E* 93 (4 Apr. 2016), p. 043313. DOI: 10.1103/PhysRevE.93.043313. URL: <https://link.aps.org/doi/10.1103/PhysRevE.93.043313> (cit. on pp. 30, 31).
- [BNT01] W. Bialek, I. Nemenman, and M. Tishby. “Predictability, complexity, and learning.” In: (2001) (cit. on p. 113).
- [Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738 (cit. on pp. 70, 71, 78).
- [Bre08] G. L. Bretthorst. “Nonuniform sampling: Bandwidth and aliasing”. In: *Concepts in Magnetic Resonance Part A* 32A.6 (2008), pp. 417–435. DOI: <https://doi.org/10.1002/cmr.a.20125>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/cmr.a.20125>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cmr.a.20125> (cit. on p. 66).
- [DB18] K. Doctor and J. Byers. “Optimal Sampling of BRDF’s of Varying Complexity”. In: July 2018, pp. 4123–4126. DOI: 10.1109/IGARSS.2018.8518762 (cit. on p. 125).
- [GC11] M. Girolami and B. Calderhead. “Riemann manifold Langevin and Hamiltonian Monte Carlo methods”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 73.2 (2011), pp. 123–214. DOI: <https://doi.org/10.1111/j.1467-9868.2010.00765.x>. eprint: <https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9868.2010.00765.x>. URL: <https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2010.00765.x> (cit. on pp. 31, 35).
- [HST01] H. Haario, E. Saksman, and J. Tamminen. “An adaptive Metropolis algorithm”. In: *Bernoulli* 7.2 (2001), pp. 223–242. DOI: [bj/1080222083](https://doi.org/10.1080/0222083010880222083). URL: [https://doi.org/](https://doi.org/10.1080/0222083010880222083) (cit. on p. 17).
- [IB02] F. S. Ilya Nemenman and W. Bialek. “Entropy and Inference, Revisited”. In: (2002) (cit. on p. 127).
- [Jay89] E. T. Jaynes. “Clearing up Mysteries — The Original Goal”. In: *Maximum Entropy and Bayesian Methods: Cambridge, England, 1988*. Ed. by J. Skilling. Dordrecht: Springer Netherlands, 1989, pp. 1–27. ISBN: 978-94-015-7860-8. DOI: 10.1007/978-94-015-7860-8_1. URL: https://doi.org/10.1007/978-94-015-7860-8_1 (cit. on p. 63).

- [KPB20] I. Kobyzev, S. Prince, and M. Brubaker. “Normalizing Flows: An Introduction and Review of Current Methods”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), pp. 1–1. ISSN: 1939-3539. DOI: 10.1109/tpami.2020.2992934. URL: <http://dx.doi.org/10.1109/TPAMI.2020.2992934> (cit. on p. 55).
- [Mac02] D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. USA: Cambridge University Press, 2002. ISBN: 0521642981 (cit. on p. 64).
- [Mac03] D. J. C. MacKay. “Information Theory, Inference, and Learning Algorithms”. In: (2003). URL: <http://www.inference.phy.cam.ac.uk/mackay/itila/> (cit. on pp. 83, 86, 97, 101).
- [PSM19] G. Papamakarios, D. C. Sterratt, and I. Murray. *Sequential Neural Likelihood: Fast Likelihood-free Inference with Autoregressive Flows*. 2019. arXiv: 1805.07226 [stat.ML] (cit. on p. 55).
- [Sei05] U. Seifert. “Entropy Production along a Stochastic Trajectory and an Integral Fluctuation Theorem”. In: *Physical Review Letters* 95.4 (July 2005). ISSN: 1079-7114. DOI: 10.1103/physrevlett.95.040602. URL: <http://dx.doi.org/10.1103/PhysRevLett.95.040602> (cit. on p. 41).
- [Vih12] M. Vihola. “Robust adaptive Metropolis algorithm with coerced acceptance rate”. In: *Statistics and Computing* 22.2 (2012). URL: <https://doi.org/10.1007/s11222-011-9269-5> (cit. on p. 19).