2

Probabilistic learning methods

Concept. To extend the formalism that we developed in the previous chapter to describe the time evolution of state probabilities. Having introduced the basic concepts, we then use this formalism to motivate some important methods for probabilistic learning. In particular, we will see how Gaussian processes regression and 'empirical probabilistic reweighting' work, as well as discuss how some phenomena permit a convenient description via 'mean-field' equations or regression of distribution moments. For the mathematically-inclined, this chapter will take a detailed look at how our formalism can be extended to focus on the time evolution of probabilities, with a view to probabilistic learning later on. For the programmers, this chapter will mostly focus on the algorithm concepts, but all of the relevant software lives in this public Git repository: https://github.com/umbralcalc/learnadex.

2.1 Probabilistic formalism

This book is about building more realistic training environments for machine learning systems in the real world. In the last chapter we formalised, designed and built a generalised simulation engine which could serve as the essential scaffolding for these environments. So why then, in this chapter, do we want to extend our formalism to talk about probabilistic learning methods?

For many of these realistic environments where only partial state observations are possible, probabilistic learning tools can immediately become an essential tool to robustly infer the state of a system and predict its future states from any given point in time. Given that these tools are often so important for extending the capability of learning algorithms to deal with partially-observed domains; we propose to extend the idea of what is more conventionally considered a 'machine learning environment' to include tools for system state inference and future state prediction. In this sense, this chapter, and the remaining chapters of **Part 1** in this book, will consider probabilistic learning as an essential part of the environment for a machine learning system.

In this chapter, we're going to discuss the formal connections between some probabilistic learning methods and the simulation formalism we introduced in the last chapter. The novelties in this chapter are less to do with the probabilistic learning methodology and lie more in the specifics of how we combine some of these ideas together when referencing the stochadex formalism, and how

this manifests in designing more generally-applicable software for the user.

Let's start by returning to the formalism that we introduced in the previous chapter. As we discussed at that point; this formalism is appropriate for sampling from nearly every stochastic phenomenon that one can think of. We are going to extend this description to consider what happens to the probability that the state history matrix takes a particular set of values over time.

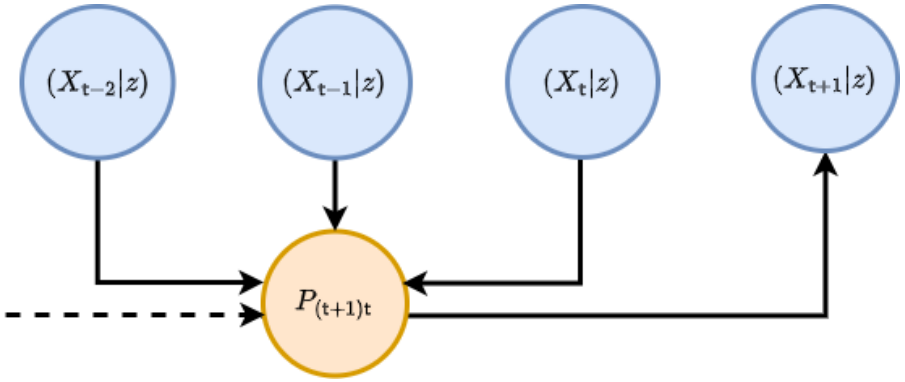

Figure 2.1: Graph representation of Eq. (2.1).

So, how do we begin? In the first chapter, we defined the general stochastic process with the formula $X^i_{t+1} = F^i_{t+1}(X_{0:t}, z, t)$. This equation also has an implicit *master equation* associated to it that fully describes the time evolution of the *probability density function* $P_{t+1}(X|z)$ of $X_{0:t+1} = X$ given that the parameters of the process are z. This can be written as

$$P_{t+1}(X|z) = P_t(X'|z)P_{(t+1)t}(x|X', z)\,, \tag{2.1}$$

where for the time being we are assuming the state space is continuous in each of the matrix elements and $P_{(t+1)t}(x|X', z)$ is the conditional probability that $X_{t+1} = x$ given that $X_{0:t} = X'$ at time t and the parameters of the process are z. To try and understand what Eq. (2.1) is saying, we find it's helpful to think of an iterative relationship between probabilities; each of which is connected by their relative conditional probabilities. We've also illustrated this kind of thinking in Fig. 2.1.

Consider what happens when we extend the chain of conditional probabilities in Eq. (2.1) back in time by one step. In doing so, we retrieve a joint probability of rows $X_{t+1} = x$ and $X_t = x'$ on the right hand side of the expression

$$P_{t+1}(X|z) = P_{t-1}(X''|z)P_{(t+1)t(t-1)}(x, x'|X'', z)\,. \tag{2.2}$$

Since Eqs. (2.1) and (2.2) are both valid ways to obtain $P_{t+1}(X|z)$ we can average between them without loss of generality in the original expression, like this

$$P_{t+1}(X|z) = \frac{1}{2}\left[P_t(X'|z)P_{(t+1)t}(x|X', z) + P_{t-1}(X''|z)P_{(t+1)t(t-1)}(x, x'|X'', z)\right]. \tag{2.3}$$

Following this line of reasoning to its natural conclusion, Eq. (2.1) can hence be generalised to consider all possible joint distributions of rows at different timesteps like this

$$P_{t+1}(X|z) = \frac{1}{t}\sum_{t''=0}^{t} P_{t''}(X''|z)P_{(t+1)t\ldots t''}(x, x', \ldots|X'', z)\,. \tag{2.4}$$

If we wanted to just look at the distribution over the latest row $X_{\mathsf{t}+1} = x$, we could achieve this through marginalisation over all of the previous matrix rows in Eq. (2.1) like this

$$P_{\mathsf{t}+1}(x|z) = \int_{\Omega_{\mathsf{t}}} \mathrm{d}X' P_{\mathsf{t}+1}(X|z) = \int_{\Omega_{\mathsf{t}}} \mathrm{d}X' P_{\mathsf{t}}(X'|z) P_{(\mathsf{t}+1)\mathsf{t}}(x|X', z)\,. \tag{2.5}$$

But what is Ω_{t}? You can think of this as just the domain of possible matrix X' inputs into the integral which will depend on the specific stochastic process we are looking at.

The symbol $\mathrm{d}X'$ in Eq. (2.5) is our shorthand notation throughout the book for computing the sum of integrals over previous state history matrices which can further be reduced via Eq. (2.4) into a product of sub-domain integrals over each matrix row

$$P_{\mathsf{t}+1}(x|z) = \frac{1}{\mathsf{t}} \sum_{\mathsf{t}''=0}^{\mathsf{t}} \left\{ \int_{\omega_{\mathsf{t}'}} \mathrm{d}^n x' ... \int_{\Omega_{\mathsf{t}''}} \mathrm{d}X'' \right\} P_{\mathsf{t}''}(X''|z) P_{(\mathsf{t}+1)\mathsf{t}...\mathsf{t}''}(x, x', ...|X'', z) \tag{2.6}$$

$$= \frac{1}{\mathsf{t}} \sum_{\mathsf{t}''=0}^{\mathsf{t}} \int_{\Omega_{\mathsf{t}''}} \mathrm{d}X'' P_{\mathsf{t}''}(X''|z) P_{(\mathsf{t}+1)\mathsf{t}''}(x|X'', z)\,, \tag{2.7}$$

where each row measure is a Cartesian product of n elements (a Lebesgue measure), i.e.,

$$\mathrm{d}^n x = \prod_{i=0}^{n} \mathrm{d}x^i\,, \tag{2.8}$$

and lowercase $x, x', \ldots$ values will always refer to individual rows within the state matrices. Note that $1/\mathsf{t}$ here is a normalisation factor — this just normalises the sum of all probabilities to 1 given that there is a sum over t'. Note also that, if the process is defined over continuous time, we would need to replace

$$\frac{1}{\mathsf{t}} \sum_{\mathsf{t}'=0}^{\mathsf{t}} \to \frac{1}{t(\mathsf{t})} \sum_{\mathsf{t}'=0}^{\mathsf{t}} \delta t(\mathsf{t}')\,. \tag{2.9}$$

Let's go through some examples. Non-Markovian phenomena with continuous state spaces can have quite complex master equations. A relatively simple example is that of pure diffusion processes which exhibit stochastic resetting at a rate r to a remembered location from the trajectory history [1]

$$P_{\mathsf{t}+1}(x|z) = (1-r) P_{\mathsf{t}}(x|z) + \sum_{i=0}^{n} \sum_{j=0}^{n} \frac{\partial}{\partial x^i} \frac{\partial}{\partial x^j} \Big[D_{\mathsf{t}}(x, z) P_{\mathsf{t}}(x|z) \Big] + r \sum_{\mathsf{t}'=0}^{\mathsf{t}} \delta t(\mathsf{t}') K[t(\mathsf{t}) - t(\mathsf{t}')] P_{\mathsf{t}'}(x|z)\,, \tag{2.10}$$

where here K is some memory kernel. For Markovian phenomena which have a continuous state space, Eqs. (2.1) and (2.5) no longer depend on timesteps older than the immediately previous one, hence, e.g., Eq. (2.5) reduces to just

$$P_{\mathsf{t}+1}(x|z) = \int_{\omega_{\mathsf{t}}} \mathrm{d}^n x'\, P_{\mathsf{t}}(x'|z) P_{(\mathsf{t}+1)\mathsf{t}}(x|x', z)\,. \tag{2.11}$$

A famous example of this kind of phenomenon arises from approximating Eq. (2.11) with an expansion (Kramers-Moyal [2, 3]) up to second-order, yielding the Fokker-Planck equation

$$P_{\mathsf{t}+1}(x|z) = P_{\mathsf{t}}(x|z) - \sum_{i=0}^{n} \frac{\partial}{\partial x^i}\left[\mu_{\mathsf{t}}(x,z)P_{\mathsf{t}}(x|z)\right] + \sum_{i=0}^{n}\sum_{j=0}^{n} \frac{\partial}{\partial x^i}\frac{\partial}{\partial x^j}\left[D_{\mathsf{t}}(x,z)P_{\mathsf{t}}(x|z)\right], \tag{2.12}$$

which describes a process undergoing drift-diffusion.

An analog of Eq. (2.5) exists for discrete state spaces as well. We just need to replace the integral with a sum and the schematic would look something like this

$$P_{\mathsf{t}+1}(x|z) = \sum_{\Omega_{\mathsf{t}}} P_{\mathsf{t}}(X'|z)P_{(\mathsf{t}+1)\mathsf{t}}(x|X',z)\,, \tag{2.13}$$

where we note that the P's in the expression above all now refer to *probability mass functions*. In what follows, discrete state space can always be considered by replacing the integrals with summations over probability masses in this manner; we only use the continuous state space formulation for our notation because one could argue it's a little more general.

Analogously to continuous state spaces, we can give some examples of master equations for phenomena with a discrete state space as well. In the Markovian case, we need look no further than a simple time-dependent Poisson process

$$P_{\mathsf{t}+1}(x|z) = \lambda(\mathsf{t})\delta t(\mathsf{t}{+}1)P_{\mathsf{t}}(x{-}1|z) + \big[1 - \lambda(\mathsf{t})\delta t(\mathsf{t}{+}1)\big]P_{\mathsf{t}}(x|z)\,. \tag{2.14}$$

For such an example of a non-Markovian system, a Hawkes process [4] master equation would look something like this

$$\begin{aligned} P_{\mathsf{t}+1}(x|z) = {} & \mu\delta t(\mathsf{t}{+}1)P_{\mathsf{t}}(x{-}1|z) + \big[1 - \mu\delta t(\mathsf{t}{+}1)\big]P_{\mathsf{t}}(x|z) \\ & + \sum_{x'=0}^{\infty}\sum_{\mathsf{t}'=0}^{\mathsf{t}} \phi[t(\mathsf{t}) - t(\mathsf{t}')]\delta t(\mathsf{t}{+}1)P_{\mathsf{t}\mathsf{t}'(\mathsf{t}'-1)}(x{-}1, x', x'{-}1|z) \\ & + \sum_{x'=0}^{\infty}\left\{1 - \sum_{\mathsf{t}'=0}^{\mathsf{t}} \phi[t(\mathsf{t}) - t(\mathsf{t}')]\delta t(\mathsf{t}{+}1)\right\}P_{\mathsf{t}\mathsf{t}'(\mathsf{t}'-1)}(x, x', x'{-}1|z)\,, \end{aligned} \tag{2.15}$$

where we note the complexity in this expression arises because it has to include a coupling between the rate at which events occur and an explicit memory of when the previous ones did occur (recorded by differencing the count between adjacent timesteps by 1).

2.2 Motivating probabilistic learning

So now that we are more familiar with the notation used by Eq. (2.5), we can use it to motivate some useful probabilistic learning methods. While it's worth going into some mathematical detail to give a better sense of where each technique comes from, we should emphasise that the methodologies we discuss here are not new to the technical literature at all. We draw on influences from Empirical Dynamical Modeling (EDM) [5], some classic nonparametric local regression techniques — such as LOWESS/Savitzky-Golay filtering [6] — and also Gaussian processes [7].

Let's begin our discussion of algorithms by integrating Eq. (2.5) over x to obtain a relation for the mean of the distribution

$$M_{\mathtt{t}+1}(z) = \int_{\omega_{\mathtt{t}+1}} \mathrm{d}^n x \, x \, P_{\mathtt{t}+1}(x|z) = \frac{1}{\mathtt{t}} \sum_{\mathtt{t}''=0}^{\mathtt{t}} \int_{\Omega_{\mathtt{t}''}} \mathrm{d}X'' P_{\mathtt{t}''}(X''|z) M_{(\mathtt{t}+1)\mathtt{t}''}(X'', z) \,, \tag{2.16}$$

where you can view the $M_{(\mathtt{t}+1)\mathtt{t}''}(X'', z)$ values as either terms in some regression model, or derivable explicitly from a known master equation. The latter of these provides one approach to statistically infer the states and parameters of stochastic simulations from data: one begins by knowing what the master equation is, uses this to compute the time evolution of the mean (and potentially higher-order statistics) and then connects these $\mathtt{t}$ and z-dependent statistics back to the likelihood of observing the data. This is what is commonly known as the 'mean-field' inference approach; averaging over the available degrees of freedom in the statistical moments of distributions. Though, knowing what the master equation is for an arbitrarily-defined stochastic phenomenon can be very difficult indeed, and the resulting equations typically require some form of approximation.

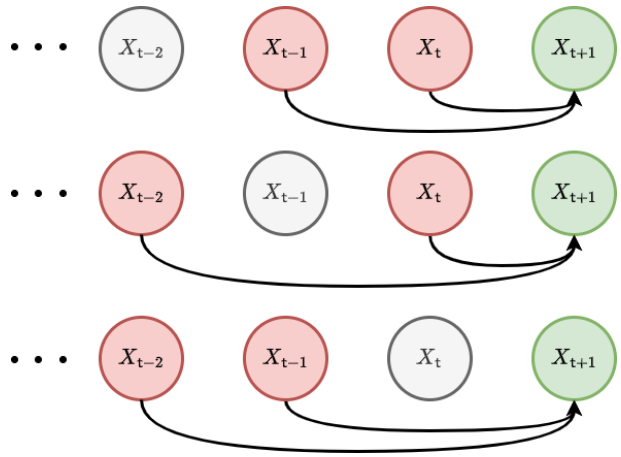

Figure 2.2: A graph representation of the correlations in Eq. (2.21).

Given that the mean-field approach isn't always going to be viable as an inference method, we should also consider other ways to describe the shape and time evolution characteristics of $P_{\mathtt{t}+1}(X|z)$. For continuous state spaces, it's possible to approximate this whole distribution with a logarithmic expansion like so

$$\ln P_{\mathtt{t}+1}(X|z) \simeq \ln P_{\mathtt{t}+1}(X_*|z) + \frac{1}{2} \sum_{\mathtt{t}'=0}^{\mathtt{t}+1} \sum_{i=0}^{n} \sum_{j=0}^{n} (x - x_*)^i \mathcal{H}^{ij}_{(\mathtt{t}+1)\mathtt{t}'}(z)(x' - x'_*)^j \tag{2.17}$$

$$\mathcal{H}^{ij}_{(\mathtt{t}+1)\mathtt{t}'}(z) = \frac{\partial}{\partial x^i} \frac{\partial}{\partial (x')^j} \ln P_{\mathtt{t}+1}(X|z) \Big|_{X=X_*} \,, \tag{2.18}$$

where the values for X_* (and its rows $x_*, x'_*, \dots$) are defined by the vanishing of the first derivative,

i.e., these are chosen such that

$$\frac{\partial}{\partial x^i} \ln P_{\mathsf{t}+1}(X|z)\Big|_{X=X_*} = 0\,. \tag{2.19}$$

This logarithmic expansion is one way to see how a Gaussian process regression is able to approximate X_t evolving in time for many different processes. By selecting the appropriate function for the kernel given by Eq. (2.18), a Gaussian process regression can be fully specified.

If we keep the truncation up to second order in Eq. (2.17), note that this expression implies a pairwise correlation structure of the form

$$P_{\mathsf{t}+1}(X|z) \to \prod_{\mathsf{t}'=0}^{\mathsf{t}} P_{(\mathsf{t}+1)\mathsf{t}'}(x, x'|z) = \prod_{\mathsf{t}'=0}^{\mathsf{t}} P_{\mathsf{t}'}(x'|z) P_{(\mathsf{t}+1)\mathsf{t}'}(x|x', z)\,. \tag{2.20}$$

Given this pairwise temporal correlation structure, Eq. (2.7) reduces to this simpler sum of integrals

$$P_{\mathsf{t}+1}(x|z) = \frac{1}{\mathsf{t}} \sum_{\mathsf{t}'=0}^{\mathsf{t}} \int_{\omega_{\mathsf{t}'}} \mathrm{d}^n x' P_{\mathsf{t}'}(x'|z) P_{(\mathsf{t}+1)\mathsf{t}'}(x|x', z)\,. \tag{2.21}$$

We have illustrated these second-order correlations with a graph visualisation in Fig. (2.2).

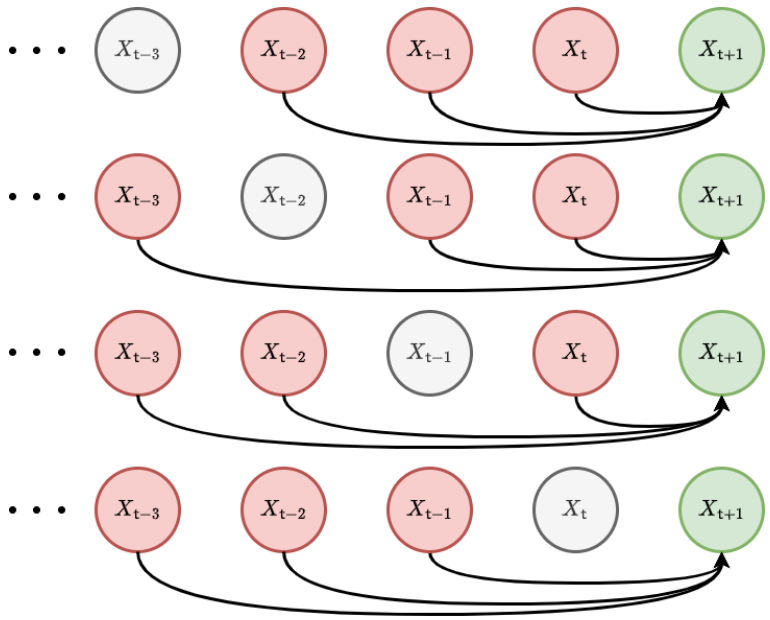

Figure 2.3: A graph representation of the correlations in Eq. (2.23).

In a similar fashion, we can increase the expansion order of Eq. (2.17) to include third-order correlations such that

$$P_{\mathsf{t}+1}(X|z) \to \prod_{\mathsf{t}'=0}^{\mathsf{t}} \prod_{\mathsf{t}''=0}^{\mathsf{t}'-1} P_{\mathsf{t}'\mathsf{t}''}(x', x''|z) P_{(\mathsf{t}+1)\mathsf{t}'\mathsf{t}''}(x|x', x'', z)\,, \tag{2.22}$$

and, in this instance, one can show that Eq. (2.7) reduces to

$$P_{\mathsf{t}+1}(x|z) = \frac{1}{\mathsf{t}}\sum_{\mathsf{t}'=0}^{\mathsf{t}} \frac{1}{\mathsf{t}'-1}\sum_{\mathsf{t}''=0}^{\mathsf{t}'-1}\int_{\omega_{\mathsf{t}'}} \mathrm{d}^n x' \int_{\omega_{\mathsf{t}''}} \mathrm{d}^n x'' P_{\mathsf{t}'\mathsf{t}''}(x', x''|z) P_{(\mathsf{t}+1)\mathsf{t}'\mathsf{t}''}(x|x', x'', z)\,. \tag{2.23}$$

We have also illustrated these third-order correlations with another graph visualisation in Fig. (2.3). Using $P_{\mathsf{t}'\mathsf{t}''}(x', x''|z) = P_{\mathsf{t}''}(x''|z)P_{\mathsf{t}'\mathsf{t}''}(x'|x'', z)$ one can also show that this integral is a marginalisation of this expression

$$P_{(\mathsf{t}+1)\mathsf{t}''}(x|x'', z) = \frac{1}{\mathsf{t}}\sum_{\mathsf{t}'=0}^{\mathsf{t}}\int_{\omega_{\mathsf{t}'}} \mathrm{d}^n x' P_{\mathsf{t}'\mathsf{t}''}(x'|x'', z) P_{(\mathsf{t}+1)\mathsf{t}'\mathsf{t}''}(x|x', x'', z)\,, \tag{2.24}$$

which describes the time evolution of the conditional probabilities. Note how this implies that the Gaussian process kernel itself can be evolved through time to replicate these higher-order temporal correlations for a regression problem, if desired.

Another probabilistic learning algorithm that we can consider is what we shall call 'empirical probabilistic reweighting'. There is another expression for the mean of the distribution, that we can derive under certain conditions, which will be valuable to motivating this algorithm. If the probability distribution over each row of the state history matrix is *stationary* — meaning that $P_{\mathsf{t}+1}(x|z) = P_{\mathsf{t}'}(x|z)$ — it's possible to go one step further than Eq. (2.16) and assert that

$$M_{\mathsf{t}+1}(z) = \int_{\omega_{\mathsf{t}+1}} \mathrm{d}^n x\, x\, P_{\mathsf{t}+1}(x|z) = \frac{1}{\mathsf{t}}\sum_{\mathsf{t}'=0}^{\mathsf{t}}\int_{\omega_{\mathsf{t}'}} \mathrm{d}^n x'\, x'\, P_{\mathsf{t}'}(x'|z)\int_{\omega_{\mathsf{t}+1}} \mathrm{d}^n x\, P_{(\mathsf{t}+1)\mathsf{t}'}(x|x', z)\,. \tag{2.25}$$

To see that Eq. (2.25) is true, first note that a joint distribution over both x and x' can be derived like this $P_{(\mathsf{t}+1)\mathsf{t}'}(x, x'|z) = P_{(\mathsf{t}+1)\mathsf{t}'}(x|x', z)P_{\mathsf{t}'}(x'|z)$. Secondly, note that this joint distribution will always allow variable swaps trivially like this $P_{(\mathsf{t}+1)\mathsf{t}'}(x, x'|z) = P_{\mathsf{t}'(\mathsf{t}+1)}(x', x|z)$. Then, lastly, note that stationarity of $P_{\mathsf{t}+1}(x|z) = P_{\mathsf{t}'}(x|z)$ means

$$\begin{aligned}
\frac{1}{\mathsf{t}}\sum_{\mathsf{t}'=0}^{\mathsf{t}}\int_{\omega_{\mathsf{t}+1}} \mathrm{d}^n x \int_{\omega_{\mathsf{t}'}} \mathrm{d}^n x'\, x\, P_{(\mathsf{t}+1)\mathsf{t}'}(x, x'|z) &= \frac{1}{\mathsf{t}}\sum_{\mathsf{t}'=0}^{\mathsf{t}}\int_{\omega_{\mathsf{t}'}} \mathrm{d}^n x \int_{\omega_{\mathsf{t}+1}} \mathrm{d}^n x'\, x\, P_{\mathsf{t}'(\mathsf{t}+1)}(x, x'|z) \\
&= \frac{1}{\mathsf{t}}\sum_{\mathsf{t}'=0}^{\mathsf{t}}\int_{\omega_{\mathsf{t}'}} \mathrm{d}^n x' \int_{\omega_{\mathsf{t}+1}} \mathrm{d}^n x\, x'\, P_{(\mathsf{t}+1)\mathsf{t}'}(x, x'|z) \\
&= \frac{1}{\mathsf{t}}\sum_{\mathsf{t}'=0}^{\mathsf{t}}\int_{\omega_{\mathsf{t}'}} \mathrm{d}^n x'\, x'\, P_{\mathsf{t}'}(x'|z)\int_{\omega_{\mathsf{t}+1}} \mathrm{d}^n x\, P_{(\mathsf{t}+1)\mathsf{t}'}(x|x', z)\,,
\end{aligned}$$

where we've used the trivial variable swap and integration variable relabelling to arrive at the second equality in the expressions above.

The standard covariance matrix elements can also be computed in a similar fashion

$$\begin{aligned}
C^{ij}_{\mathsf{t}+1}(z) &= \int_{\omega_{\mathsf{t}+1}} \mathrm{d}^n x\, [x - M_{\mathsf{t}+1}(z)]^i [x - M_{\mathsf{t}+1}(z)]^j P_{\mathsf{t}+1}(x|z) \\
&= \frac{1}{\mathsf{t}}\sum_{\mathsf{t}'=0}^{\mathsf{t}}\int_{\omega_{\mathsf{t}'}} \mathrm{d}^n x'\, [x' - M_{\mathsf{t}+1}(z)]^i [x' - M_{\mathsf{t}+1}(z)]^j\, P_{\mathsf{t}'}(x'|z)\int_{\omega_{\mathsf{t}+1}} \mathrm{d}^n x\, P_{(\mathsf{t}+1)\mathsf{t}'}(x|x', z)\,.
\end{aligned} \tag{2.26}$$

While they look quite abstract, Eqs. (2.25) and (2.26) express the core idea behind how the probabilistic reweighting will function. By assuming a stationary distribution, we gain the ability to directly estimate the statistics of the probability distribution of the next sample from the stochastic process $P_{\mathsf{t}+1}(x|z)$ from past samples it may have in empirical data; which are represented here by $P_{\mathsf{t}'}(x'|z)$. More on this later.

2.3 Learning algorithms

Probabilistic reweighting depends on the stationarity of $P_{\mathsf{t}+1}(x|z) = P_{\mathsf{t}'}(x|z)$ such that, e.g., Eq. (2.25) is applicable. The core idea behind it is to represent the past distribution of state values $P_{\mathsf{t}'}(x'|z)$ with the samples from a real time series dataset. If the user then specifies a good model for the relationships in this data by providing a weighting function which returns the conditional probability mass

$$\mathsf{w}_{\mathsf{t}'}(y,z) = \int_{\omega_{\mathsf{t}+1}} \mathrm{d}^n x \, P_{(\mathsf{t}+1)\mathsf{t}'}(x|x'{=}y,z)\,, \tag{2.27}$$

we can apply this as a *reweighting* of the historical time series samples to estimate any statistics of interest. Taking Eqs. (2.25) and (2.26) as the examples; we are essentially approximating these integrals through weighted sample estimations like this

$$M_{\mathsf{t}+1}(z) \simeq \frac{1}{\mathsf{t}} \sum_{\mathsf{t}'=0}^{\mathsf{t}} Y_{\mathsf{t}'} \mathsf{w}_{\mathsf{t}'}(Y_{\mathsf{t}'}, z) \tag{2.28}$$

$$C^{ij}_{\mathsf{t}+1}(z) \simeq \frac{1}{\mathsf{t}} \sum_{\mathsf{t}'=0}^{\mathsf{t}} [Y_{\mathsf{t}'} - M_{\mathsf{t}+1}(z)]^i [Y_{\mathsf{t}'} - M_{\mathsf{t}+1}(z)]^j \, \mathsf{w}_{\mathsf{t}'}(Y_{\mathsf{t}'}, z)\,, \tag{2.29}$$

where we have defined the data matrix Y with rows $Y_{\mathsf{t}+1}, Y_{\mathsf{t}}, \ldots$, each of which representing specific observations of the rows in X at each point in time from a real dataset.

The goal of a learning algorithm for probabilistic reweighting would be to learn the optimal reweighting function $\mathsf{w}_{\mathsf{t}'}(Y_{\mathsf{t}'}, z)$ with respect to z, i.e., the ones which most accurately represent a provided dataset. But before we think about the various kinds of conditional probability we could use, we need to think about how to connect the post-reweighting statistics to the data by defining an objective function.

If the mean is a sufficient statistic for the distribution which describes the data, a choice of, e.g., Exponential, Poisson or Binomial distribution could be used where the mean is estimated directly from the time series using Eq. (2.25), given a conditional probability $P_{(\mathsf{t}+1)\mathsf{t}'}(x|x', z)$. Extending this idea further to include distributions which also require a variance to be known, e.g., the Normal, Gamma or Negative Binomial distributions could be used where the variance (and/or covariance) could be estimated using Eq. (2.26). These are just a few simple examples of distributions that can link the estimated statistics from Eqs. (2.25) and (2.26) to a time series dataset. However, the algorithmic framework is very general to whatever choice of 'data linking' distribution that a researcher might need.

We should probably make what we've just said a little more mathematically concrete. We can define $P_{\mathsf{t}+1}[y; M_{\mathsf{t}+1}(z), C_{\mathsf{t}+1}(z), \ldots]$ as representing the likelihood of $y = Y_{\mathsf{t}+1}$ given the estimated statistics from Eqs. (2.25) and (2.26) (and maybe higher-orders). Note that in order to do this, we

need to identify the x' and $\mathtt{t}'$ values that are used to estimate, e.g., $M_{\mathtt{t}+1}(z)$ with the past data values which are observed in the dataset time series itself. Now that we have this likelihood, we can immediately evaluate an objective function (a cumulative log-likelihood) that we might seek to optimise over for a given dataset

$$\ln \mathcal{L}_{\mathtt{t}+1}(Y|z) = \sum_{\mathtt{t}'=0}^{\mathtt{t}+1} \ln P_{\mathtt{t}'}[y; M_{\mathtt{t}'}(z), C_{\mathtt{t}'}(z), \dots], \tag{2.30}$$

where the summation continues until all of the past measurements $Y_{\mathtt{t}+1}, Y_{\mathtt{t}}, \dots$ which exist as rows in the data matrix Y have been taken into account. The code to compute this objective function follows the schematic we have provided in Fig. 2.4.

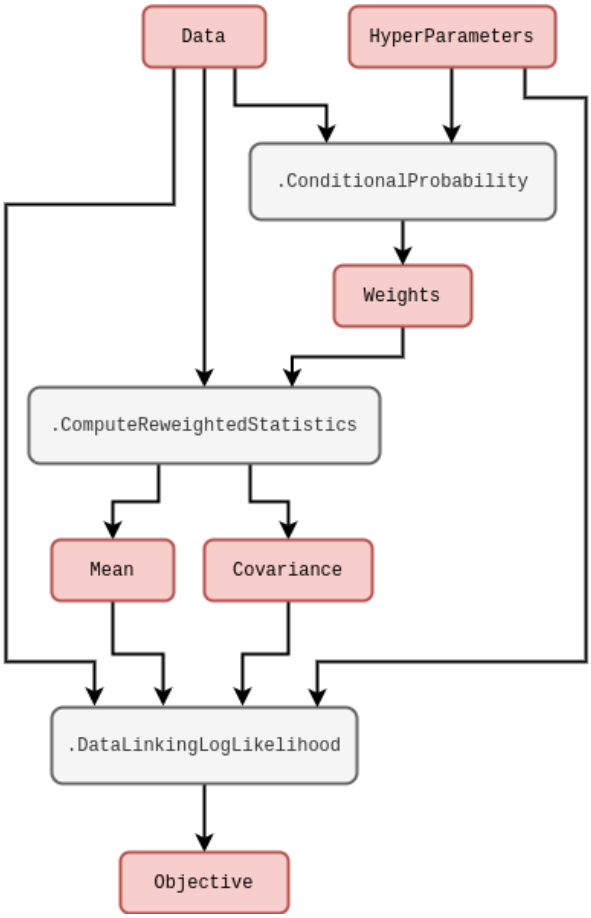

Figure 2.4: Code schematic of the probability reweighting objective computation.

In order to specify what $P_{(\mathtt{t}+1)\mathtt{t}'}(x|x', z)$ is, it's quite natural to define a set of hyperparameters for the elements of z. To get a sense of how the data-linking function relates to these hyperparameters, it's instructive to consider an example. One generally-applicable option for the conditional

probability could be a purely time-dependent kernel

$$P_{(\mathsf{t}+1)\mathsf{t}'}(x|x',z) \propto \mathcal{K}(z,\mathsf{t}+1,\mathsf{t}')\,, \tag{2.31}$$

and the data-linking distribution, e.g., could be a Gaussian

$$P_{\mathsf{t}+1}[y; M_{\mathsf{t}+1}(z), C_{\mathsf{t}+1}(z), \dots] = \mathsf{MultivariateNormalPDF}[y; M_{\mathsf{t}+1}(z), C_{\mathsf{t}+1}(z)]\,. \tag{2.32}$$

It's worth pointing out that other machine learning frameworks could easily be used to model these conditional probabilities. For example, neural networks could be used to infer the optimal reweighting scheme and this would still allow us to use the data-linking distribution.[1] It would still be desirable to keep the data-linking distribution as it can usually be sampled from very easily — something that can be quite difficult to achieve with a purely machine learning-based representation of the distribution. Sampling itself could even be made more flexible by leveraging a Variational Autoencoder (VAE) [9]; these use neural networks not just on the compression (or 'encode') step to estimate the statistics but also use them as a layer between the sample from the data distribution model and the output (the 'decode' step).

In the case of Eqs. (2.31) and (2.32) above, the hyperparameters that would be optimised could relate to the kernel in a wide variety of ways. Optimising them would make our optimised reweighting similar to (but very much *not* the same as) evaluating maximum a posteriori (MAP) of a Gaussian process regression. In a Gaussian process regression, one is concerned with inferring the the whole of X_{t} as a function of time using the pairwise correlations implied by Eq. (2.17). Based on this expression, the cumulative log-likelihood for a Gaussian process can be calculated as follows

$$\ln \mathcal{L}_{\mathsf{t}+1}(Y|z) = -\frac{n}{2}\ln(2\pi) - \frac{1}{2}\ln|\mathcal{H}(z)| - \frac{1}{2}\sum_{\mathsf{t}'=0}^{\mathsf{t}+1}\sum_{i=0}^{n}\sum_{j=0}^{n} y^i \mathcal{H}^{ij}_{(\mathsf{t}+1)\mathsf{t}'}(z)(y')^j\,. \tag{2.33}$$

As we did for the reweighting algorithm, in Fig. 2.5 we have illustrated a schematic of the basic code needed to compute the objective function of a learning algorithm based on Eq. (2.33). Note that, in the expression above, we have replaced $x - x_*$ with y and so it is assumed that the data has already been shifted such that its values are positioned around the distribution peak. Knowing where this peak will be a priori is not possible. However, for Gaussian data, an unbiased estimator for this peak will be the sample mean and so we have included an initial data standardisation in the steps outlined by Fig. 2.5.

The optimisation approach that we choose to use for obtaining the best hyperparameters in the conditional probability of Eq. (2.30) will depend on a few factors. For example, if the number of hyperparameters is relatively low, but their gradients are difficult to calculate exactly; then a gradient-free optimiser (such as the Nelder-Mead [10] method or something like a particle swarm [11, 12]) would likely be the most effective choice. On the other hand, when the number of hyperparameters ends up being relatively large, it's usually quite desriable to utilise the gradients in algorithms like vanilla Stochastic Gradient Descent [13] (SGD) or Adam [14].

If the gradients of Eq. (2.30) are needed, we can always factorise each derivative with respect

[1] One can think of using this neural network-based reweighting scheme as similar to constructing a normalising flow model [8] with an autoregressive layer. Invertibility and further network structural constraints mean that these are not exactly equivalent, however.

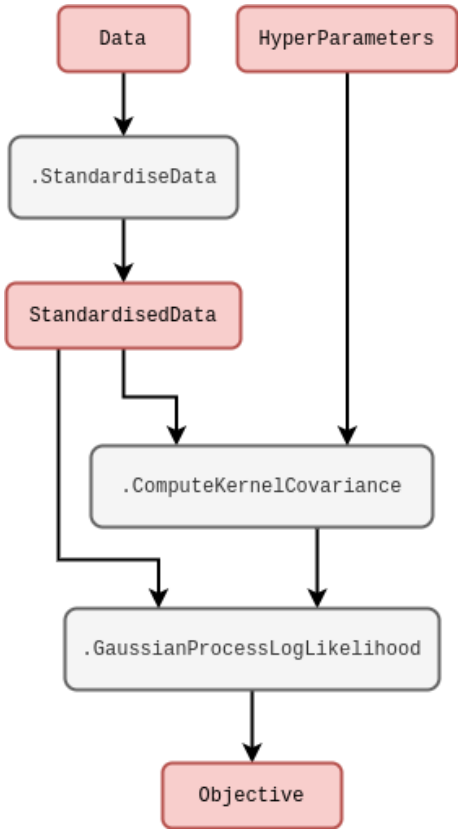

Figure 2.5: Code schematic of the Gaussian process objective computation.

to hyperparameter z^i in the following way through the chain rule

$$\frac{\partial}{\partial z^i} \ln \mathcal{L}_{t+1}(Y|z) = \sum_{t'=0}^{t+1} \frac{\partial M_{t'}}{\partial z^i} \frac{\partial}{\partial M_{t'}} \ln P_{t'}[y; M_{t'}(z), C_{t'}(z), \dots] + \sum_{t'=0}^{t+1} \frac{\partial C_{t'}}{\partial z^i} \frac{\partial}{\partial C_{t'}} \ln P_{t'}[y; M_{t'}(z), C_{t'}(z), \dots] . \tag{2.34}$$

By factoring derivatives in this manner, the computation can be separated into two parts: the derivatives with respect to $M_{t'}$ and $C_{t'}$, which are typically quite straightforward; and the derivatives with respect to z elements, which typically need a more involved calculation depending on the model. Incidentally, this separation also neatly lends itself to abstracting gradient calculations as having a simpler, general purpose component that can be built directly into a library of data models and a more complex, model-specific component that the user must specify.

The same logic should apply to a learning algorithm which optimises z to obtain the MAP for a Gaussian process. If the gradients of Eq. (2.33) are required, the user would have to specify

derivatives of the kernel matrix (and its determinant) with respect to z in order to calculate

$$\frac{\partial}{\partial z^k} \ln \mathcal{L}_{\mathfrak{t}+1}(Y|z) = -\frac{1}{2} \frac{\partial}{\partial z^k} \ln |\mathcal{H}(z)| - \frac{1}{2} \sum_{\mathfrak{t}'=0}^{\mathfrak{t}+1} \sum_{i=0}^{n} \sum_{j=0}^{n} y^i \frac{\partial}{\partial z^k} \mathcal{H}^{ij}_{(\mathfrak{t}+1)\mathfrak{t}'}(z)(y')^j \,. \tag{2.35}$$

Bibliography

[1] D. Boyer, M. R. Evans, and S. N. Majumdar, "Long time scaling behaviour for diffusion with resetting and memory," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2017, no. 2, p. 023208, 2017.

[2] H. A. Kramers, "Brownian motion in a field of force and the diffusion model of chemical reactions," *Physica*, vol. 7, no. 4, pp. 284–304, 1940.

[3] J. Moyal, "Stochastic processes and statistical physics," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 11, no. 2, pp. 150–210, 1949.

[4] A. G. Hawkes, "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, vol. 58, no. 1, pp. 83–90, 1971.

[5] G. Sugihara and R. M. May, "Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series," *Nature*, vol. 344, no. 6268, pp. 734–741, 1990.

[6] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[7] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[8] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 11, pp. 3964–3979, 2020.

[9] L. Pinheiro Cinelli, M. Araújo Marins, E. A. Barros da Silva, and S. Lima Netto, "Variational autoencoder," in *Variational Methods for Machine Learning with Applications to Deep Networks*. Springer, 2021, pp. 111–149.

[10] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[12] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*. IEEE, 1998, pp. 69–73.

[13] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.