4

Online simulation inference

Concept. To generalise the procedure of statistical inference for any simulation model using an algorithm which builds from techniques we developed in the previous chapter. When we say 'statistical inference' here; we specifically mean computing the maximum a posteriori (MAP) estimate for any arbitrary stochastic model which has been defined in the stochadex simulator. For the mathematically-inclined, this chapter will give a very brief exposition for Bayesian statistical inference metholodology — in particular, how it relates to the evaluation of MAP estimates. For the programmers, the software described in this chapter lives in the public Git repository: https://github.com/umbralcalc/learnadex.

4.1 Inference formalism

In Bayesian inference, one applies Bayes' rule to the problem of statistically inferring a model from some dataset. This typically involves the following formula for a posterior distribution

$$\mathcal{P}_{\mathsf{t}+1}(z|Y) \propto \mathcal{L}_{\mathsf{t}+1}(Y|z)\mathcal{P}(z)\,. \tag{4.1}$$

In the formula above, one relates the prior probability distribution over a parameter set $\mathcal{P}(z)$ and the likelihood $\mathcal{L}_{\mathsf{t}+1}(Y|z)$ of some data matrix Y up to timestep $\mathsf{t}+1$ given the parameters z of a model to the posterior probability distribution of parameters given the data $\mathcal{P}_{\mathsf{t}+1}(z|Y)$ up to some proportionality constant. All this may sound a bit technical in statistical language, so it can also be helpful to summarise what the formula above states verbally as follows: the initial (prior) state of knowledge about the parameters z we want to learn can be updated by some likelihood function of the data to give a new state of knowledge about the values for z (the 'posterior' probability).

From the point of view of statistical inference, if we seek to maximise $\mathcal{P}_{\mathsf{t}+1}(z|Y)$ — or its logarithm — in Eq. (4.1) with respect to z, we will obtain what is known as a maximum posteriori (MAP) estimate of the parameters. In fact, we have already encountered this metholodology in the previous chapter when discussing the algorithm which obtains the best fit parameters for the empirical probability reweighting. In this case; while it appears that we optimised the log-likelihood

directly as our objective function, one can easily show that this is also technically equivalent obtaining a MAP estimate where one chooses a specfic prior $\mathcal{P}(z) \propto 1$ (typically known as a 'flat prior').

How might we calulate the posterior in practice with some arbitrary stochastic process model that has been defined in the stochadex? In order to make the comparison to a real dataset, any stochadex model of interest will always need to be able to generate observations which can be directly compared to the data. To formalise this a little; a stochadex model could be represented as a map from z to a set of stochastic measurements $\mathsf{Y}_{\mathsf{t}+1}(z), \mathsf{Y}_{\mathsf{t}}(z), \dots$ that are directly comparable to the rows in the real data matrix Y. The values in Y may only represent a noisy or partial measurement of the latent states of the simulation X, so a more complete picture can be provided by the following probabilistic relation

$$P_{\mathsf{t}+1}(\mathsf{y}|z) = \int_{\omega_{\mathsf{t}+1}} \mathrm{d}^n x \, P_{\mathsf{t}+1}(\mathsf{y}|x) P_{\mathsf{t}+1}(x|z) \,, \tag{4.2}$$

where, in practical terms, the measurement probability $P_{\mathsf{t}+1}(\mathsf{y}|x)$ of $\mathsf{Y}_{\mathsf{t}+1} = \mathsf{y}$ given $X_{\mathsf{t}+1} = x$ can be represented by sampling from another stochastic process which takes the state of the stochadex simulation as input. Given that we have this capability to compare like-for-like between the data and the simulation; the next problem is to figure out how this comparison between two sequences of vectors can be done in a way which ensures the the statistics of the posterior are ultimately respected.

For an arbitrary simulation model which is defined by the stochadex, the likelihood in Eq. (4.1) is typically not describable as a simple function or distribution. While we could train the probability reweighting we derived in the previous chapter to match the simulation; to do this well would require having an exact formula for the conditional probability, and this is not always easy to derive in the general case. Instead, there is a class of Bayesian inference methods which we shall lean on to help us compute the posterior distribution (and hence the MAP), which are known as 'Likelihood-Free' methods [1, 2, 3, 4].

'Likelihood-Free' methods work by separating out the components of the posterior which relate to the closeness of rows in Y to the rows in Y from the components which relate the states X and parameters z of the simulation stochastically to Y. To achieve this separation, we can make use of chaining conditional probability like this

$$\mathcal{P}_{\mathsf{t}+1}(X, z|Y) = \int_{\Upsilon_{\mathsf{t}+1}} \mathrm{d}\mathsf{Y} \, \mathcal{P}_{\mathsf{t}+1}(\mathsf{Y}|Y) P_{\mathsf{t}+1}(X, z|\mathsf{Y}) \,, \tag{4.3}$$

where $\Upsilon_{\mathsf{t}+1}$ here corresponds to the domain of the simulated measurements matrix Y at time $\mathsf{t}+1$.

As we demonstrated in the previous chapter, it's possible for us to also optimise a probability distribution $\mathcal{P}_{\mathsf{t}'}(\mathsf{y}|Y) = P_{\mathsf{t}'}(\mathsf{y}; \mathcal{M}_{\mathsf{t}'}, \mathcal{C}_{\mathsf{t}'}, \dots)$ for each step in time to match the statistics of the measurements in Y as well as possible, given some statistics $\mathcal{M}_{\mathsf{t}'} = \mathcal{M}_{\mathsf{t}'}(Y)$ and $\mathcal{C}_{\mathsf{t}'} = \mathcal{C}_{\mathsf{t}'}(Y)$. Assuming the independence of samples (rows) in Y, this distribution can be used to construct the distribution over all of Y through the following product

$$\mathcal{P}_{\mathsf{t}+1}(\mathsf{Y}|Y) = \prod_{\mathsf{t}'=0}^{\mathsf{t}+1} P_{\mathsf{t}'}(\mathsf{y}; \mathcal{M}_{\mathsf{t}'}, \mathcal{C}_{\mathsf{t}'}, \dots) \,. \tag{4.4}$$

We do not necessarily need to obtain these statistics from the probability reweighting method, but could instead try to fit them via some other objective function. Either way, this represents a lossy

compression of the data we want to fit the simulation to, and so the best possible fit is desirable; regardless of overfitting. This choice to summarise the data with statistics means we are using what is known as a Bayesian Synthetic Likelihood (BSL) method [2, 3] instead of another class of methods which approximate an objective function directly using a proximity kernel — known as Approximate Bayesian Computation (ABC) methods [1].

Let's consider a few concrete examples of $P_{\mathsf{t}'}(\mathsf{y}; \mathcal{M}_{\mathsf{t}'}, \mathcal{C}_{\mathsf{t}'}, \dots)$. If the data measurements were well-described by a multivariate normal distribution, then

$$P_{\mathsf{t}'}(\mathsf{y}; \mathcal{M}_{\mathsf{t}'}, \mathcal{C}_{\mathsf{t}'}, \dots) = \mathsf{MultivariateNormalPDF}(\mathsf{y}; \mathcal{M}_{\mathsf{t}'}, \mathcal{C}_{\mathsf{t}'}) \,, \tag{4.5}$$

Similarly, if the data measurements were instead better described by a Poisson distribution, we might disregard the need for a covariance matrix statistic $\mathcal{C}_{\mathsf{t}'}$ and instead use

$$P_{\mathsf{t}'}(\mathsf{y}; \mathcal{M}_{\mathsf{t}'}, \mathcal{C}_{\mathsf{t}'}, \dots) = \mathsf{PoissonPMF}(\mathsf{y}; \mathcal{M}_{\mathsf{t}'}) \,. \tag{4.6}$$

The more statistically-inclined readers may notice that the probability mass function here would require the integrals in Eq. (4.3) to be replaced with summations over the relevant domains.

Eq. (4.3) demonstrates how one can construct a statistically meaningful way to compare the sequence of real data measurements $Y_{\mathsf{t}+1}, Y_{\mathsf{t}}, \dots$ to their modelled equivalents $\mathsf{Y}_{\mathsf{t}+1}(z), \mathsf{Y}_{\mathsf{t}}(z), \dots$. But we still haven't shown how to compute $P_{\mathsf{t}+1}(X, z|\mathsf{Y})$ for a given simulation, and this can be the most challenging part. To begin with, we can reapply Bayes' rule and the chaining of conditional probability to find

$$P_{\mathsf{t}+1}(x, z|\mathsf{Y}) \propto P_{\mathsf{t}+1}(\mathsf{y}|z) P_{\mathsf{t}}(z|\mathsf{Y}') = P_{\mathsf{t}+1}(\mathsf{y}|x) P_{\mathsf{t}+1}(x|z) P_{\mathsf{t}}(z|\mathsf{Y}') \,, \tag{4.7}$$

where here $P_{\mathsf{t}}(z|\mathsf{Y}')$ is the probability of $\mathsf{Y}_{\mathsf{t}} = \mathsf{Y}'$.

The relationship between $P_{\mathsf{t}+1}(X|z)$ and previous timesteps can be directly inferred from the probabilistic iteration formula that we introduced in the previous chapter. So we can map probabilities of $X_{0:\mathsf{t}+1} = X$ throughout time and learned information about the state of the system can be applied from previous values, given z. But is there a similar relationship we might consider for $P_{\mathsf{t}+1}(z|\mathsf{Y})$? Yes there is! The marginalisation

$$P_{\mathsf{t}+1}(z|\mathsf{Y}) \propto \left[\int_{\Omega_{\mathsf{t}+1}} \mathrm{d}^n x \, P_{\mathsf{t}+1}(\mathsf{y}|x) P_{\mathsf{t}+1}(x|z) \right] P_{\mathsf{t}}(z|\mathsf{Y}') \,, \tag{4.8}$$

shows how the z updates can occur in an iterative fashion. The reader may also recognize the factor above in brackets as Eq. (4.2). To complete the picture, one can combine the X and z updates into a joint distribution update which takes the following form

$$P_{\mathsf{t}+1}(X, z|\mathsf{Y}) \propto P_{\mathsf{t}+1}(\mathsf{y}|x) P_{(\mathsf{t}+1)\mathsf{t}}(x|X', z) P_{\mathsf{t}}(X', z|\mathsf{Y}') \,. \tag{4.9}$$

We can also marginalise this distribution over the past state history rows to get a distribution over the latest state row $X_{\mathsf{t}+1} = x$ like this

$$P_{\mathsf{t}+1}(x, z|\mathsf{Y}) = \int_{\Omega_{\mathsf{t}}} \mathrm{d}X' P_{\mathsf{t}+1}(X, z|\mathsf{Y}) \propto P_{\mathsf{t}+1}(\mathsf{y}|x) \int_{\Omega_{\mathsf{t}}} \mathrm{d}X' P_{(\mathsf{t}+1)\mathsf{t}}(x|X', z) P_{\mathsf{t}}(X', z|\mathsf{Y}') \,. \tag{4.10}$$

In the next section, we're going to discuss how to translate all of this probabilistic language into some MAP inference algorithms. Before we do this, however, it will be instructive (particularly for

'online' learning algorithms) to consider what happens if the model changes over time and z needs to change in order to better represent the real data. In such situations, we propose to apply the same formula as Eq. (4.10) but instead replace the distribution over (X', z) on the right hand side with its 'past discounted' version[1]

$$\int_{\Omega_{\mathsf{t}}} \mathrm{d}X' P_{\mathsf{t}}(X', z|\mathsf{Y}') \;\longrightarrow\; \frac{1}{\mathsf{t}} \sum_{\mathsf{t}'=0}^{\mathsf{t}} \int_{\omega_{\mathsf{t}'}} \mathrm{d}^n x' \beta^{\mathsf{t}-\mathsf{t}'} P_{\mathsf{t}'}(x', z|\mathsf{Y}')\,, \tag{4.11}$$

where $0 < \beta < 1$ and we recall the notation which considers distributions over the individual rows x' within the matrix X' in this new version. This time-dependent discount factor could be used to reduce the dependence of the update on data which is much further in the past, and hence will ultimately lead to a more responsive algorithm. This responsiveness would have to be balanced with the tradeoffs associated with discounting potentially valuable data that may offer greater long-term stability. Readers who are familiar with reinforcement learning may be starting to feel in familiar territory here — they will have to wait for the latter parts of the book to see more on discounting though!

4.2 Online learning the MAP

Eq. (4.9) tells us how to probabilistically translate the current state of knowledge about (x, z) forward through time in response to the arrival of new data. We also know how to connect the simulated measurements to the real data because Eq. (4.3) essentially gives us an objective function to maximise for each step in time. This is all great in theory; but in practice, this optimisation problem typically has several layers of difficulty to it. Since the model has been defined by its stochastically generated samples of measurements $\mathsf{Y}_{\mathsf{t}+1}(z), \mathsf{Y}_{\mathsf{t}}(z), \ldots$, the objective function will manifestly be stochastic too. Another layer of difficulty is that gradients of the objective function are not immediately computable and so navigation around the optimisation domain could be difficult, especially in high-dimensional problems. Lastly, given that the simulation model in the stochadex needs to be running multiple times for each timestep, we need a way of mitigating computational expense.

So how should we proceed? To solve this problem in the general case, Eqs. (4.3) and (4.9) tell us we need to synthesize the following components into a single algorithm:

1. A process $P_{(\mathsf{t}+1)\mathsf{t}}(x|X', z)$ which iterates the state matrix of the simulation X forward in time.
2. A process $P_{\mathsf{t}+1}(\mathsf{y}|x)$ which generates a simulated measurement from the simulated state x.
3. A probability distribution $P_{\mathsf{t}'}(\mathsf{y}; \mathcal{M}_{\mathsf{t}'}, \mathcal{C}_{\mathsf{t}'}, \ldots)$ which represents the posterior distribution of the simulated measurement vector y given an optimised compression of the real data into summary statistics.
4. Some way of representing samples from the distribution $P_{\mathsf{t}+1}(X, z|\mathsf{Y})$ so that their distribution can be updated and will coverge towards the posterior over (X, z).

[1] In the continuous-time version, this past-discounting factor can depend on the stepsize such that we replace

$$\beta^{\mathsf{t}-\mathsf{t}'} \longrightarrow \frac{1}{\beta[\delta t(\mathsf{t})]} \prod_{\mathsf{t}''=\mathsf{t}'}^{\mathsf{t}} \beta[\delta t(\mathsf{t}'')]\,.$$

Cite this nice paper which outlines all the recent kinds of simulation inference: [5].

Keep the heuristic Bayes posterior estimator method as it is an example of recursive Bayes estimation - it can also be used to filter the ensemble at every step to make a particle filter [6]. Amortize this online learning process by training a neural net to produce the best estimates for the filter from the input real data!

Before writing this up, should read this paper on efficient amortized inference using neural networks with `BayesFlow` here in particular: [7]. But also, should cite other works to make amortized inference more efficient by using neural networks to learn convenient functions of the Bayes factor in Evidence networks [8].

- amortized online inference of the posterior update over just z can be achieved by running lots of simulations and solving the inverse problem with the `y` outputs i.e., neural network modelling of the update in Eq. (4.8)

The algorithm is specifically: 1. if this is a refit step, sample new values for (X, z) for all members of the ensemble from the current (X, z) distribution points and run the iterations for all of these ensemble members from the back of the window all the way up to the current point in time (hence the full matrix X is sampled) 2. take all of the ensemble members a step forward in time 3. approximate the mode by computing the average values of z within the q-th percentile of the sampled probability mass (where q is set by the user and is ideally $< 68\%$) — this idea comes from nested sampling 4. stream in the data for the next point in time and go to 1.

As such an algorithm converges, we can recompute (and hence iteratively improve) the MAP estimate with respect to each iteration of the posterior.

Readers with some machine learning experience may be familiar with the classic exploration vs exploitation tradeoffs. It's clear that these tradeoffs will manifest in our case here when trying to strike a balance between iterating the posterior distribution and optimizing the current posterior with respect to (X, z) to compute the MAP.

Readers of the previous section may also have recognized that Eq. (4.9) contains the same conditional probability $P_{(\mathrm{t}+1)\mathrm{t}}(x|X', z)$ as the reweighting algorithm. This structure enables us to reuse all of the exposition we provided for the probabilistic reweighting and highlights how the reweighting itself can be used in the algorithm to optimise the posterior.

If we now synthesize both of these observations together, we can see how a stochastic variant of the well-known Expectation-Maximisation Algorithm [9, 10, 11] naturally emerges.

4.3 Software design

Take a step back at this point and consider all the use-cases for the learnadex:

- Core functionality is to enable iterative updates to the $P_{\mathrm{t}}(X, z)$ distribution at every timestep. There must also be flexibility in how this distribution can represented, i.e., either:
  - a set of Monte Carlo samples, or
  - a set of distribution parameters
- For Monte Carlo samples, we must also keep the flexibility to use either a BSL or ABC-style data-to-sim comparison in order to facilitate the update

- For distribution parameters, the update can be custom-built by the user with online likelihood-based inference/Bayes estimator methods
- Separate thread context runs of online learning methods using an update method, e.g.,
  - Gradient-free batch optimisation with any chosen learning algorithm
  - Gradient-based parameter updates
  - Arbitrary parameter updates from a different method

Bibliography

[1] S. A. Sisson, Y. Fan, and M. Beaumont, *Handbook of approximate Bayesian computation*. CRC Press, 2018.

[2] L. F. Price, C. C. Drovandi, A. Lee, and D. J. Nott, "Bayesian synthetic likelihood," *Journal of Computational and Graphical Statistics*, vol. 27, no. 1, pp. 1–11, 2018.

[3] S. N. Wood, "Statistical inference for noisy nonlinear ecological dynamic systems," *Nature*, vol. 466, no. 7310, pp. 1102–1104, 2010.

[4] C. Drovandi and D. T. Frazier, "A comparison of likelihood-free methods with and without summary statistics," *Statistics and Computing*, vol. 32, no. 3, p. 42, 2022.

[5] K. Cranmer, J. Brehmer, and G. Louppe, "The frontier of simulation-based inference," *Proceedings of the National Academy of Sciences*, vol. 117, no. 48, pp. 30 055–30 062, 2020.

[6] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on signal processing*, vol. 50, no. 2, pp. 174–188, 2002.

[7] S. T. Radev, U. K. Mertens, A. Voss, L. Ardizzone, and U. Köthe, "Bayesflow: Learning complex stochastic models with invertible neural networks," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 4, pp. 1452–1466, 2020.

[8] N. Jeffrey and B. D. Wandelt, "Evidence networks: simple losses for fast, amortized, neural bayesian model comparison," *arXiv preprint arXiv:2305.11241*, 2023.

[9] H. O. Hartley, "Maximum likelihood estimation from incomplete data," *Biometrics*, vol. 14, no. 2, pp. 174–194, 1958.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society: series B (methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

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