Bayesian Abstraction of Markov Population Models

Luca Bortolussi[1,2] and Francesca Cairoli[1]

[1] DMG, University of Trieste, Italy
[2] MOSI, Saarland University, Germany lbortolussi@units.it | francesca.cairoli@phd.units.it

Abstract. Markov Population Models are a widespread formalism, with applications in Systems Biology, Performance Evaluation, Ecology, and many other fields. The associated Markov stochastic process in continuous time is often analyzed by simulation, which can be costly for large or stiff systems, particularly when simulations have to be performed in a multi-scale model (e.g. simulating individual cells in a tissue). A strategy to reduce computational load is to abstract the population model, replacing it with a simpler stochastic model, faster to simulate. Here we pursue this idea, building on previous work [3] and constructing an approximate kernel for a Markov process in continuous space and discrete time, capturing the evolution at fixed Δt time steps. This kernel is learned automatically from simulations of the original model. Differently form [3], which relies on deep neural networks, we explore here a Bayesian density regression approach based on Dirichlet processes, which provides a principled way to estimate uncertainty.

Keywords: Model abstraction · Markov Population Models · Bayesian density regression · Dirichlet processes.

1 Introduction

Stochastic models are undoubtedly one of the most powerful frameworks to describe and reason about complex systems. Due to the severe state space explosion of these models, simulation is often the only viable tool to analyse them. Even simulation, however, can face severe computational limits, in particular when the systems of interest have a multi-scale nature. Consider for example a biological scenario, in which we want to model the effect of a drug targeting individual cells in a tissue, for instance a tumour. In order to build an accurate model of such a system, both the dynamics at the individual cells and the one at the tissue level have to be described and simulated. Unfortunately, tissues typically contain millions of cells, each requiring the simulation of complex interaction pathways [5]. This complexity defies also modern High Performance Computing resources, and can be tackled only by simplifying the model of each individual cell, i.e. resorting to model abstraction.

Typical approaches in this direction require a large dose of experience and ingenuity to hand-craft a suitable abstraction. Recent alternatives rely on modern

artificial intelligence to learn the best abstraction from a given class of models. The more general the class, the more flexible the method, the higher the learning cost.

Related Work Some approaches in literature try to introduce as much knowledge in the abstraction as possible, thus reducing the complexity of the learning problem. A notable example is [13], in which authors exploit knowledge about the key drivers of bacterial chemotaxis, combining an abstract model of the dynamic of such drivers with a simple model with few states describing the decision of the bacteria, i.e. whether to rotate or proceed straight. The final model is a Continuous Time Markov Chain (CTMC) where transition rates are learned using Gaussian Process Regression from some simulations of the full model. On the other hand of the spectrum, we find the work of Palaniappan et al. [16], in which the authors start from a bunch of simulations of the original model, using information theoretic ideas extract a subset of relevant variables, discretize them and then learn a dynamic Bayesian network in discrete time. The abstract model was used for fast approximate simulation of the original model. The work on this paper follows these lines, starting from the approach of [3], in which we abstracted a CTMC model by discretizing time, choosing a time step Δt relevant for the dynamics of the higher organisational scale (e.g. the time-scale of the diffusion dynamics at the tissue level). The so obtained Discrete Time Markov Process is defined in continuous space, approximating the exact transition matrix by a transition kernel modelled as a mixture of Gaussian distributions with means and covariances taken as functions of the current state of the model. These functions are learned using Deep Neural Networks. Despite the method was effective in the examples studied, it is not without drawbacks. First of all, there is no quantification of the uncertainty in the so-learned kernel, hence no measure of confidence on the accuracy of the abstraction. Secondly, the number of mixture components is a hyperparameter strongly affecting the performance of the method.

Contributions In this work, we continue the investigation started in [3], exploring the use of non-parametric Bayesian machine leaning [1] to provide a consistent estimate of uncertainty and to self-tune kernel complexity from data.

Our idea is to work with probability distributions in the space of transition kernels, defining a prior distribution and computing a posterior by conditioning on observed simulation data. If we fix the current state $\mathbf{x}$ in the model, distributions over kernels reduce to distributions over probability distributions of the next state $\mathbf{x}'$ after Δt units of time. To simulate the abstract model, we first need to sample a distribution, and then sample the next state from this distribution. Provided these sampling operations can be implemented efficiently, this could considerably speed-up simulation. Importantly, having a distribution of distributions allows us to incorporate uncertainty in the reconstruction of the kernel and identify initial states from which the reconstruction is more problematic. This

information can be used to tune the accuracy of the abstraction, e.g. improve it in areas of the state space visited more often by the process.

The technical details of this approach are non-trivial. Fortunately, the problem we want to solve has been studied in statistics and machine learning and goes under the name of density regression [7, 6]. Borrowing from this literature, we extend and tailor existing methods that use Dirichlet processes (essentially distributions over probability densities) to finally obtain a posterior distribution which is a mixture of Gaussian with a variable number of components, learned from data. The posterior distribution cannot be computed analytically, hence we rely on Gibbs sampling, a Monte Carlo method which has to be run only at training time. In fact, during training we will sample from the posterior and approximate it by its empirical distribution which permits a very fast sampling from the transition kernel. The method is validated on few experimental case studies, which we use to discuss potentials and limitations of this approach.

Paper structure In Section 2 we introduce some background material on stochastic models and case studies. Section 3 is devoted to present the model abstraction framework and formulate the learning problems, while density regression and the algorithm we use is described in detail in Section 4. Experiments are reported in Section 5, while the final discussion is in Section 6.

2 Background

2.1 Chemical Reaction Networks

Chemical Reaction Networks (CRNs) use the formalism of chemical equations to capture the dynamics of population models, including biological systems and epidemic spreading scenarios. Let $X_1, \dots, X_m$ be a collection of m species and $\eta_{t,i}$, $i = 1, \dots, m$, denotes the population size of species X_i present in the system at time t. The dynamics of a CRN is described by a set of reactions $\mathcal{R} = \{R_1, \dots, R_p\}$. The firing of reaction R_i results in a transition of the system from state $\eta_t = (\eta_{t,1}, \dots, \eta_{t,m}) \in \mathcal{S} = \mathbb{N}^m$ to state $\eta_t + \nu_i$, with ν_i being the update vector. A general reaction R_i is identified by the tuple (f_{R_i}, ν_i), where f_{R_i}, known as propensity function of reaction R_i, depends on the state of the system.

The time evolution of a CRN can be modelled as a Continuous Time Markov Chain (CTMC [14]) on the discrete space $\mathcal{S}$. Motivated by the well-known memoryless property of CTMC, let $\mathbb{P}_{s_0}(\eta_t = s)$ denote the probability of finding the system in state s at time t given that it was in state s_0 at time t_0. This probability satisfies a system of ODEs known as Chemical Master Equation (CME):

$$\partial_t \mathbb{P}_{s_0}(\eta_t = s) = \sum_{j=1}^{p} \left[f_{R_j}(s - \nu_j) \mathbb{P}_{s_0}(\eta_t = s - \nu_j) - f_{R_j}(s) \mathbb{P}_{s_0}(\eta_t = s) \right]. \quad (1)$$

Since the CME is a system in general with countably many differential equations, its analytic or numeric solution is almost always unfeasible. An alternative computational approach is to generate trajectories using stochastic algorithms for simulation, like the well-known the Gillespie's SSA [10].

The SIR model The SIR epidemiological model describes a population of N individuals divided in three mutually exclusive groups: susceptible (S), infected (I) and recovered (R). The system state at time t is $\eta_t = (S_t, I_t, R_t)$. The possible reactions, given by the interaction of individuals (representing the molecules of a CRN), are the following:

- $R_1 : S + I \xrightarrow{k_1 \cdot I_t S_t / N} 2I$ (infection),
- $R_2 : I \xrightarrow{k_2 \cdot I_t}$ R (recovery).

The model describes the spread, in a population with fixed size N, of an infectious disease that grants immunity to those who recover from it. As the SIR model is well-known and stable, we use it as a testing ground for our Bayesian abstraction procedure.

The Gene Regulatory Network model Consider a simple self-regulated gene network [2] in which a single gene G is transcribed to produce copies of a mRNA molecule M; each mRNA molecule can then be translated into a protein P. In addition P acts as a repressor with respect to the gene G. In other words, the gene activity is regulated through a negative-feedback loop, a common pattern in biological systems. The reactions are the following:

- $R_1 : G^{ON} \xrightarrow{k_{prodM} \cdot G^{ON}} G^{ON} + M$ (transcription)
- $R_2 : M \xrightarrow{k_{prodP} \cdot M} M + P$ (translation)
- $R_3 : G^{ON} \xrightarrow{k_{deact} \cdot G^{ON}} G^{OFF}$ (protein binding)
- $R_4 : G^{OFF} \xrightarrow{k_{act} \cdot G^{OFF}} G^{ON} + P$ (protein unbinding)
- $R_5 : M \xrightarrow{k_{degM} \cdot M} \varnothing$ (mRNA degradation)
- $R_6 : P \xrightarrow{k_{degP} \cdot P} \varnothing$ (protein degradation).

The system dynamic varies significantly accordingly with the choice of reaction rates. We refer to [2] for a detailed exploration of different behavioural patterns. According to our choice, see Section 5, the system exhibits several well-separated stable configurations. If we look at trajectories in Figure 1 on a smaller scale, we notice that each stable point is actually noisy, i.e. there is a high number of small amplitude oscillations. Starting from an initial state η_0, we can sequentially generate a large number of instances of the state of the system at a fixed time time t_1, η_{t_1}. The empirical probability distribution of such η_{t_1} approximates, for a number of samples sufficiently large, the density function $\mathbb{P}_{\eta_0}(\eta_t)$, shown in Figure 3 (blue lines). The system exhibits up to 5 distinguishable modes, with some Gaussian noise affecting each one of them.

Motivated by the multimodality of this system we seek a Bayesian nonparametric density regression method, able to capture such hierarchical structure.

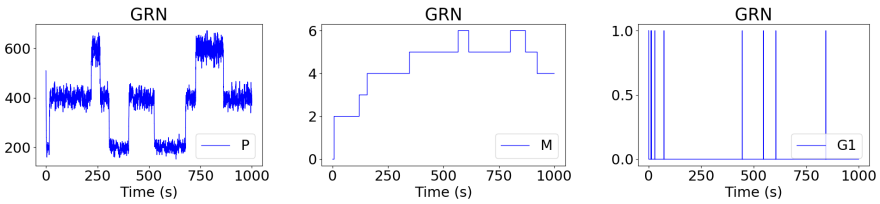

Fig. 1. Example of a trajectory, for the gene regulation network model. The overall time is 10,000 seconds. Left: trajectory of proteins (P), which present a multistable configuration. Middle: trajectory for the mRNA molecules (M). Right: trajectory for the active gene G^{ON}.

3 Model Abstraction

Let's now introduce the concept of model abstraction as presented in [3]. The underlying idea is the following: given a stochastic process $\{\eta_t\}_{t\geq 0}$ with transition probabilities $\mathbb{P}_{s_0}(\eta_t = s)$, we aim at finding another stochastic process whose trajectories are similar to the first one, but faster to simulate. Instead of working with transition probabilities themselves we rather use transition kernels. This requires a discretization of time. In other words, the process is considered only at time points with a fixed temporal distance. If we fix a time step Δt and an initial time instant $t_0 \in \mathbb{R}$, the states can be expressed as $\tilde{\eta}_i := \eta_{t_0+i\Delta t}$. The CTMC, $\{\eta_t\}_{t\geq 0}$, is now expressed as a time-homogeneous Discrete Time Markov Chain $\{\tilde{\eta}_i\}_i$ with transition kernel

$$K_d(s \mid s_0) = \mathbb{P}_{s_0}(\eta_{\Delta t} = s), \tag{2}$$

for all $s, s_0 \in \mathcal{S}$. Two additional approximations are required:

1. The abstract model takes values in $\mathcal{S}' = \mathbb{R}^m_{\geq 0}$, a continuous space in which the state space $\mathcal{S} = \mathbb{N}^m$ is embedded.
2. The kernel K_d, equation 2, is approximated by a new kernel $K(s' \mid s'_0)$ taking values in the continuous space $\mathcal{S}'$.

In constructing the approximate kernel $K(s' \mid s'_0)$, rather than trying to preserve the full behavior of the process, we restrict our attention to a time-bounded *reward function* r from $\mathcal{S}^M$ to an arbitrary space $\mathcal{T}$ (i.e. $\mathbb{R}$, $\mathbb{N}$, $\mathbb{B}$, or $\mathbb{R}^k$). Here M is an upper bound on the duration of discrete time trajectories we consider to evaluate the reward; we indicate time-bounded trajectories by $\tilde{\eta}_{[0,M]}$. Such a function r can be a projection, monitoring the number of molecules belonging to a certain subset of chemical species at a certain time step, or it can take Boolean values in $\mathbb{B} = \{0, 1\}$, representing the truth of a linear temporal property, for example checking if the system has entered into a dangerous region. Note that $r(\tilde{\eta}_{[0,M]})$ is a probability distribution on $\mathcal{T}$. The formal definition of an abstract model is the following.

Definition 1. Let $\eta = \{\eta_i\}_{i=0}^M$ be a discrete time stochastic process over an arbitrary state space $\mathcal{S}$, with $M \in \mathbb{N}_+$ a time horizon, and let $r : \mathcal{S}^M \to \mathcal{T}$ be the associated reward function. An abstraction of (η, r) is a triple $(\mathcal{S}', p, r', \eta' = \{\eta'_i\}_{i=0}^M)$ where:

- $\mathcal{S}'$ is the *abstract state space*;
- $p : \mathcal{S} \to \mathcal{S}'$ is the *abstraction function*;
- $r' : \mathcal{S}'^M \to T$ is the *abstract reward*;
- $\eta' = \{\eta'_i\}_{i=0}^M$ is the *abstract discrete time stochastic process* over $\mathcal{S}'$.

Definition 2. Let $\varepsilon > 0$, η' is said to be ε-close to η with respect to d if, for almost any $s_0 \in \mathcal{S}$,

$$d\big(r(\eta_{[0,M]}), r'(\eta'_{[0,M]})\big) < \varepsilon \qquad \text{conditioned on } \eta_0 = s_0,\ \eta'_0 = p(s_0). \tag{3}$$

Dataset Generation The model abstraction procedure can be translated into a supervised learning problem. Choose n random initial states $\{s_0^{(j)}\}_{j=1}^n$ from $\mathcal{S}$. Starting from each of these states we run a simulation from t_0 to $t_1 := t_0 + \Delta t$. $\eta_{t_1}^{(j)}$ denotes the system state at time t_1 for each one of these simulations. By defining $\mathbf{x}_j := s_0^{(j)}$ and $\mathbf{y}_j := \eta_{t_1}^{(j)}$ for all $j \in \{1, \dots, n\}$, we have thus built a dataset $\mathcal{D} := \{(\mathbf{x}_j, \mathbf{y}_j\}_{j=1}^n$, where each $\mathbf{y}_j$ is a sample from the probability distribution $\mathbb{P}_{\mathbf{x}_j}(\eta_{\Delta t})$.

In order to validate the abstraction procedure, we choose a high number of different initial settings, different from the initial states of the training set, and from them a very large number of SSA trajectories is simulated. The empirical distribution obtained can be compared with the distribution estimated with the abstract kernel at these points.

Abstract Model Simulation We now have an abstract model that can be used to simulate a trajectory. We just need to sample up to time horizon $M > 0$ from the approximate kernel K, starting from the initial state s_0 and initial time t_0. The simulated trajectory lies on the continuous state space $\mathcal{S}'$. Each time step of our simulations has thus a fixed computational cost that does not depend on the Δt chosen. This saves a lot of computational resources when simulating long trajectories. This algorithm can be easily employed in a multi-scale setting: we just need to train the kernel of the abstract model once, but after that a high number of simulations can be performed at a very high speed.

Measuring the error The error introduced by the abstract model, i.e. how much the abstract distribution differs from $r(\eta'_{[0,M]})$, is a fundamental ingredient to quantify. In general, the distance among two random distributions, X and Y, can be computed using the L_1 norm. In practice, this metric will be evaluated statistically, resulting in the so called histogram distance [4]

$$D(X, Y) = \sum_{i=1}^{K} \frac{|h_X(I_i) - h_Y(I_i)|L}{K}, \tag{4}$$

where $I_1, \ldots, I_K$ are K bins of size L and $h_*(I_i)$ indicates the number of samples in bin I_i. We call *self distance* the histogram distance between two set of samples drawn from the same distribution.

Confidence intervals: The big advantage of keeping a Bayesian non-parametric perspective is the possibility to estimate the uncertainty in the approximation procedure. The transition kernel is estimated using a trace of predictive densities, from which we can reconstruct a confidence interval. We expect this interval to be tight when the abstract kernel well approximate the original kernel, and wide when the reconstruction is poor.

4 Bayesian Density Regression

Density estimation [9] is a well known process to model the density from which a given set of observations is drawn. If data are assumed to be distributed hierarchically, meaning each point belongs to a randomly chosen cluster and members of a cluster are further distributed randomly within that cluster, we are dealing with a data clustering problem. In order to place a prior probability on the structure of data, we may assume, for instance, that there are K normally distributed clusters, each cluster with its own parameters. The Bayesian non-parametric intuition is to work without pre-specifying the number of clusters K and select instead a random prior over an infinite set of clusters with infinitely many parameters. Dirichlet processes [8], which are the infinite-dimensional generalization of Dirichlet distributions, are used as prior on such unknown distribution. They are denoted as $DP(\alpha G_0)$, where α is the precision parameter and G_0 is the base measure. A parametric form, Gaussian in our case, with unknown parameters is usually chosen for G_0. Realizations of such process, $G \sim DP(\alpha G_0)$, are random distributions. In order to fit the model based on data, we should compute the posterior distribution over cluster probabilities and their associated parameters. Since we cannot write the posterior explicitly, we are going to draw samples from the posterior using a Gibbs sampling algorithm.

However, our problem, which is embedded in the supervised learning scenario, as presented in Section 3, is to estimate the conditional distribution of a variable $y \in \mathcal{Y}$, a one dimensional projection of the state space $\mathcal{S}$, that depends on a vector of covariates $\mathbf{x} \in \mathcal{X}$, which represents the entire state space $\mathcal{S}$. This task is called conditional density estimation or *density regression*. In other words, given n observations $\mathcal{D} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \in \mathcal{X} \times \mathcal{Y}$, we would like to estimate, for a generic $\mathbf{x} \in \mathcal{X}$, the density on $\mathcal{Y}$ of the response variable y, i.e. the conditional density $f(y|\mathbf{x})$.

A simple solution under the assumption that $G \sim DP(\alpha G_0)$, would be to use the dependent DP approach of MacEachern [11], which relies on the stick-breaking representation [17] of DP:

$$G = \sum_{h=1}^{\infty} \pi_h \cdot \delta_{\theta_h}, \tag{5}$$

with $\pi_h / \prod_{l=1}^{h-1}(1-\pi_l) \sim Beta(1,\alpha)$. In the formula above, $\pi = (\pi_h, h = 1,\dots,\infty)$ is an infinite sequence of stick breaking weights, δ_θ is a degenerate distribution with all its mass at θ and $\theta = (\theta_h, h = 1,\dots,\infty)$ are atoms sampled from G_0. In the form of a DP mixture of normal linear regression models, the stick-breaking process of (5) becomes

$$f(y_i|\mathbf{x}_i) = \sum_{h=1}^{\infty} \pi_h \cdot \mathcal{N}(y_i|\mathbf{x}_i\beta_h, \tau_h^{-1}), \tag{6}$$

where $\theta_h = (\beta_h, \tau_h)$ is still sampled from G_0.

The challenge is that unexpected changes in the shape of the density depending on the predictor values $\mathbf{x}$ may occur, therefore we cannot assume the distribution G to be constant over $\mathcal{X}$. In this more complex scenario, priors for a collection of dependent random distributions, $G_\mathbf{x}$ with $\mathbf{x} \in \mathcal{X}$, must be considered. Dunson, Pillai and Park [7] proposed a kernel-weighted mixture of DPs (WMDP), using a non-parametric mixture of linear regression models for the conditional density of y given $\mathbf{x}$. The conditional density function is expressed as a mixture of parametric densities:

$$f(y|\mathbf{x}) = \int_\Phi f(y|\mathbf{x},\phi) \cdot dG_\mathbf{x}(\phi), \tag{7}$$

where $f(y|\mathbf{x},\phi)$ is a known density on $\mathcal{Y}$ that depends on a parameter $\phi \in \Phi$ and $G_\mathbf{x}$ is a random mixing distribution on Φ indexed by the predictor $\mathbf{x} \in \mathcal{X}$. The unknown collection of mixture distributions is allowed to vary with predictors by defining a WMDP prior. See [7] for a detailed treatment.

Since mixtures of a sufficiently large number of Gaussian distributions have been proved to be able to approximate any distribution accurately, we focus on the following mixture of regression models:

$$f(y_i|\mathbf{x}_i) = \int \mathcal{N}(y_i|\mathbf{x}_i'\beta_i, \tau_i^{-1}) \cdot dG_\mathbf{x}(\phi_i), \tag{8}$$

with $\phi_i = (\beta_i, \tau_i)$. In order to limit the number of clusters, the WMDP prior proposed in [7] set restrictions on the uncountable collection of mixture distributions $G_\mathcal{X} = \{G_\mathbf{x} : \mathbf{x} \in \mathcal{X}\}$. For every $\mathbf{x} \in \mathcal{X}$, they express $G_\mathbf{x}$ as

$$G_\mathbf{x} = \sum_{i=1}^{n} \pi_i(\mathbf{x}) G^*_{\mathbf{x}_i}, \qquad G^*_{\mathbf{x}_i} \sim DP(\alpha G_0), \tag{9}$$

where $\pi(\mathbf{x}) = [\pi_1(\mathbf{x}),\dots,\pi_n(\mathbf{x})]$ is a vector of probability weights with $\sum_i \pi_i(\mathbf{x}) = 1$. This formulation introduces independent DP random basis distributions at each of the predictor values in the sample, and then mixes across these basis distributions to obtain a prior for the unknown mixture distribution, $G_\mathbf{x}$, at each possible predictor value, $\mathbf{x} \in \mathcal{X}$. Suppose that $(\phi_i|\mathbf{x}_i) \sim G_{\mathbf{x}_i}$, for $i = 1,\dots,n$, then, by marginalizing out the infinite-dimensional WMDP prior, it is possible

to obtain a generalization of the so called DP Polya urn scheme:

$$(\phi_i|\phi^{(i)}, X, \alpha) = \left(\frac{\alpha}{\alpha + w_i}\right) G_0 + \sum_{j \neq i} \left(\frac{w_{ij}}{\alpha + w_i}\right) \delta_{\phi_j}, \tag{10}$$

where w_{ij} are weights that depend on the function π, the DP parameter α and the set of observed predictors $\mathbf{X}$, and $w_i = \sum_{j \neq i} w_{ij}$. In [6] the explicit specification of π was avoided, using a simpler and more interpretable form: $w_{ij} = w_\psi(\mathbf{x}_i, \mathbf{x}_j)$, where $w_\psi : \mathcal{X} \times \mathcal{X} \to [0, 1]$ is a bounded kernel measuring how close two predictors are in terms of a distance measure d, with ψ a smoothing parameter controlling how rapidly $w_\psi(\mathbf{x}, \mathbf{x}') \to 0$ as $d(\mathbf{x}, \mathbf{x}')$ decreases. In the limit as $\psi \to \infty$, $w_\psi(\mathbf{x}, \mathbf{x}') = 0$ for any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ having $d(\mathbf{x}, \mathbf{x}') > 0$. In addition, for all finite ψ, $\lim_{\mathbf{x} \to \mathbf{x}'} w_\psi(\mathbf{x}, \mathbf{x}') = 1$. Under this simplification, equation (10) can be written as

$$(\phi_i|\phi^{(i)}, X, \alpha, \psi) = \left(\frac{\alpha}{\alpha + w_i(\psi)}\right) G_0 + \sum_{h=1}^{k^{(i)}} \left(\frac{w^*_{ij}(\psi)}{\alpha + w_i(\psi)}\right) \delta^{(i)}_{\theta_h}, \tag{11}$$

where $\theta^{(i)} = (\theta^{(i)}_1, \ldots, \theta^{(i)}_{k^{(i)}})$ denotes the unique values of $\phi^{(i)}$ and $w^*_{ij}(\psi) = \sum_{j \neq i} 1_{(\phi_j = \theta^{(i)}_h)} w_\psi(\mathbf{x}_i, \mathbf{x}_j)$ and $w_i(\psi) = \sum_{j \neq i} w_\psi(\mathbf{x}_i, \mathbf{x}_j)$. The prior in (11) automatically allocates the n subjects into $k \leq n$ clusters according to their ϕ_i values. Because subjects located close together are more likely to be clustered together, the prior tends to penalize changes across $\mathcal{X}$ in the values of parameters. The hyperparameters α and ψ control the speed at which the prior introduce new clusters as n increase: new clusters are added more rapidly as α increases and ψ decreases.

A natural choice for w_ψ, at least for continuous $\mathbf{x}$, is the Gaussian kernel $w_\psi(\mathbf{x}, \mathbf{x}') = \exp(-\psi||\mathbf{x} - \mathbf{x}'||^2_2)$. Note that, with this kernel choice, it is important to standardize data, avoiding sensitivity to scales.

4.1 Posterior computation

Following [7], the posterior distribution needs to be computed in order to integrate out the latent cluster parameters. Since the posterior distribution is not known explicitly, but the conditional distribution of each cluster variables is known, the Gibbs sampling algorithm results being an efficient technique to estimate the posterior. Let θ be a vector of length k containing the parameter values of each of the k Gaussians distributions in the mixture, i.e. $\theta_h = (\beta_h, \tau_h)$ for $h = 1, \ldots, k$. The vector $\mathbf{S} = (S_1, \ldots, S_n)$ maps each subject to the cluster it is allocated to. In other words, $S_i = h$ if $\phi_i = \theta_h$. Excluding the i-th subject, $\theta^{(i)}$ denotes the $k^{(i)}$ unique values of $\phi^{(i)}$ and $\mathbf{S}^{(i)}$ denotes the configuration of subjects $\{1, \ldots, n\} \setminus i$ to these values.

The full conditional posterior distribution of ϕ_i is

$$(\phi_i|\phi^{(i)}, \mathcal{D}, \alpha, \psi, \gamma) \propto q_{i,0} G_{i,0} + \sum_{h=1}^{k^{(i)}} q_{i,h} \delta_{\theta^{(i)}_h}. \tag{12}$$

In the formula above, $G_{i,0}$ is the posterior obtained by updating the prior $G_0(\phi|\gamma)$, where γ indicates the hyperparameters of the base measure G_0, with the likelihood $f(y_i|\mathbf{x}_i, \phi)$:

$$G_{i,0}(\phi) = \frac{G_0(\phi|\gamma)f(y_i|\mathbf{x}_i,\phi)}{\int f(y_i|\mathbf{x}_i,\phi)dG_0(\phi|\gamma)} := \frac{G_0(\phi|\gamma)f(y_i|\mathbf{x}_i,\phi)}{h_i(y_i|\mathbf{x}_i,\gamma)}. \tag{13}$$

In addition, $q_{i,0} = c\alpha h_i(y_i|\mathbf{x}_i,\gamma)$ and $q_{i,h} = cw^*_{ih}(\psi)f(y_i|\mathbf{x}_i,\theta_h)$, where c is the normalization constant. Note that α and ψ only appear in the expressions for the configuration probabilities $q_{i,h}$.

Conditional on α and ψ, posterior computation can proceed via a Gibbs sampling algorithm, which alternates between the following three steps.

(1) Updating the configuration of subjects to clusters, $\mathbf{S}$, and the number of clusters, k, by sequentially sampling from the full conditional posterior distribution of each S_i: $\mathbb{P}(S_i = h|\phi^{(i)}, \mathcal{D}) = q_{i,h}$, for $h = 0, 1, \ldots, k$. When $S_i = 0$ a new cluster is generated sampling from from $G_{i,0}$.
(2) Updating the cluster-specific parameters θ by sampling from the full conditional posterior given the configuration, i.e. $\mathbf{S}$ and k:

$$(\theta_h|\theta^{(h)}, \mathbf{S}, k, \mathcal{D}) \propto \left(\prod_{i:S_i=h} f(y_i|x_i,\theta_h)\right) G_0(\theta_h). \tag{14}$$

(3) Updating the hyperparameters γ by sampling from their full conditional posteriors.

4.2 Implementation

As already expressed in equation (8), we are considering the case in which $f(y_i|\mathbf{x}_i,\phi_i) = \mathcal{N}(y_i|\mathbf{x}_i'\beta_i, \tau_i^{-1})$, with $\phi_i = (\beta_i', \tau_i)'$ and $\beta_i = (\beta_{i1}, \ldots, \beta_{id})'$, so that both the regression coefficients and variance can vary across clusters. This generalizes [7] where only the mean parameter was allowed to vary, while τ was kept constant. For this particular choice, the posteriors distributions, needed at step (1), (2) and (3) of the Gibbs sampler, have simple closed forms. The detailed derivation of the following equations is described in the Appendix.

A natural choice for G_0 is the multivariate normal-gamma density

$$G_0(\beta_h, \tau_h) = \mathcal{NG}(\beta_h, \tau_h|\beta, \Sigma_\beta, a_\tau, b_\tau) = \mathcal{N}(\beta_h|\beta, \tau_h^{-1}\Sigma_\beta) \cdot \mathcal{G}(\tau_h|a_\tau, b_\tau).$$

Let $\gamma = \{\beta, \Sigma_\beta, a_\tau, b_\tau\}$ denote the set of hyperparameters. In order to provide more flexibility, we allow uncertainty in γ by choosing hyper-prior densities for β, Σ_β and b_τ, while fixing a_τ. More precisely, we choose multivariate normal prior for β, $p(\beta) = \mathcal{N}(\beta|\beta_0, \Sigma_0)$, a multivariate inverse-gamma prior for Σ_β, $p(\Sigma_\beta) = \mathcal{IW}(\Sigma_\beta|\nu_0, V_0)$, also known as inverse Wishart distribution with ν_0 degrees of freedom and mean V_0, and, finally, a gamma prior for b_τ, $p(b_\tau) = \mathcal{G}(b_\tau|a_0, b_0)$.

In step (1) of the Gibbs sampler we need to compute $h_i(y_i|\mathbf{x}_i, \gamma)$, which takes the following simple form:

$$h_i(y_i|\mathbf{x}_i, \beta, \Sigma_\beta, a_\tau, b_\tau) = \frac{C(a_\tau, b_\tau) \cdot \det \Sigma_\beta}{\sqrt{2\pi} C(\bar{a}_i, \bar{b}_i) \cdot \det \bar{\Sigma}_i},$$

where $\bar{\Sigma}_i = \mathbf{x}_i\mathbf{x}_i' + \Sigma_\beta^{-1}$, $\bar{a}_i = a_\tau + \frac{1}{2}.$, $\bar{b}_i = b_\tau + \frac{1}{2}[(y_i - \mathbf{x}_i'\bar{\beta}_i)^2 + (\bar{\beta}_i - \beta)'\Sigma_\beta^{-1}(\bar{\beta}_i - \beta)]$ with $\bar{\beta}_i = \bar{\Sigma}_i^{-1}(\mathbf{x}_i y_i + \Sigma_\beta^{-1}\beta)$ and $C(a, b) = \frac{b^a}{\Gamma(a)}$, where $\Gamma(a)$ is the gamma function.

In addition, the full conditional posterior distribution of $\theta_h = (\beta_h', \tau_h)$, required at step (2), is

$$(\beta_h, \tau_h|\theta^{(h)}, \mathbf{S}, k, \gamma, \mathcal{D}) \sim \mathcal{N}_d(\beta_h|\tilde{\beta}, \tau_h^{-1}\tilde{\Sigma}_h^{-1}) \cdot \mathcal{G}(\tau_h|a_\tau + \frac{n_h}{2}, b_\tau + \frac{\xi_h^2}{2}).$$

If we denote with $\mathbf{X}_h$ and $\mathbf{Y}_h$ the vectors containing the values of predictors and responses for the n_h subjects in cluster h, the terms $\tilde{\Sigma}_h$, $\tilde{\beta}_h$ and ξ_h^2, in the formula above, are defined as follow: $\tilde{\Sigma}_h = (\mathbf{X}_h'\mathbf{X}_h + \Sigma_\beta^{-1})$, $\tilde{\beta}_h = \tilde{\Sigma}_h^{-1}(\mathbf{X}_h' \cdot \mathbf{Y}_h + \Sigma_\beta^{-1}\beta)$ and $\xi_h^2 = (\mathbf{Y}_h - \mathbf{X}_h\tilde{\beta}_h)' \cdot (\mathbf{Y}_h - \mathbf{X}_h\tilde{\beta}_h) + (\tilde{\beta}_h - \beta)' \cdot \Sigma_\beta^{-1} \cdot (\tilde{\beta}_h - \beta)$.

Finally, the conditional posterior distributions of hyper-parameters $(\beta, \Sigma_\beta, b_\tau)$, needed at step (3), are defined as follows.

The posterior for β is $(\beta|\theta, \beta_0, \Sigma_0) \sim \mathcal{N}(\beta|\hat{\beta}, \hat{\Sigma}^{-1})$ where $\hat{\Sigma} = (\sum_h \tau_h) \cdot \Sigma_\beta^{-1} + \Sigma_0^{-1}$ and $\hat{\beta} = \hat{\Sigma}^{-1}(\Sigma_0^{-1}\beta_0 + \sum_h \tau_h \Sigma_\beta^{-1}\beta_h)$.

The posterior for Σ_β is $(\Sigma_\beta|\beta_1, \ldots, \beta_k) \sim \mathcal{IW}(\Sigma_\beta|\eta_0 + k, V_0 + S)$, where $S = \sum_h \tau_h \cdot (\beta_h - \beta)(\beta_h - \beta)'$.

The b_τ posterior is $(b_\tau|\tau_1, \ldots, \tau_k) \sim \mathcal{G}(b_\tau|\tilde{a}, \tilde{b})$, where $\tilde{a} = a_0 + k \cdot a_\tau$ and $\tilde{b} = b_0 + \sum_h \tau_h$.

4.3 Conditional predictive density

Once the posterior distribution has been computed, we can finally estimate the response density for new subjects $\mathbf{x}_* \in \mathcal{X}$, i.e. $f(y_*|\mathbf{x}_*)$. This can be done using the simple form of the conditional predictive density:

$$\begin{aligned} f(y_*|\mathbf{x}_*, Y, X, \mathbf{S}, k, \theta, \gamma, \alpha, \psi) = &\left(\frac{\alpha}{\alpha + w_*(\psi)}\right) h_*(y_*|\mathbf{x}_*, \gamma) + \\ &+ \left(\frac{1}{\alpha + w_*(\psi)}\right) \sum_{h=1}^{k} \bar{w}_{*,h}(\psi) \mathcal{N}(y_*|\mathbf{x}_*'\beta_h, \tau_h^{-1}), \end{aligned} \tag{15}$$

where $w_*(\psi) = \sum_{i=1}^n w_{*,i}(\psi)$ and $\bar{w}_{*,h} = \sum_{i=1}^n 1_{(S_i=h)} w_{*,i}(\psi)$. In words, this means that each normal component has a weight that depends on the number of subjects in the dataset allocated to that component and on the cumulative distance from these subjects and $\mathbf{x}_*$. Instead, if α is large or only few subjects in the dataset are close to $\mathbf{x}_*$, we will observe a shrinkage towards the first component.

Measuring uncertainty in the approximation After convergence, each iteration t of the Gibbs sampler correspond to a mixture of Gaussians distributions, linear in $\mathbf{x}_*$, with parameters $\mathbf{S}^{(t)}, k^{(t)}, \theta^{(t)}, \gamma^{(t)}$. This density can be computed, using (15), for a dense grid of possible y_* values, obtaining a graphical representation of the mixture density. One can apply this procedure for a large number, T, of iterates after convergence of the Gibbs sampler, ending up with a trace of T predictive density distributions. From this trace, one can calculate the expected predictive density averaging over the large number of iterates. This remove also the conditioning on $\mathbf{S}, k, \theta, \gamma$. In practice, given $t = 1, \dots, T$ we obtain the estimator:

$$\hat{f}(y_*|\mathbf{x}_*, \mathcal{D}, \alpha, \psi) = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{\alpha}{\alpha + w_*(\psi)}\right) h_*(y_*|\mathbf{x}_*, \gamma^{(t)}) + \tag{16}$$

$$+ \left(\frac{1}{\alpha + w_*(\psi)}\right)\sum_{h=1}^{k} \bar{w}_{*,h}(\psi)\mathcal{N}(y_*|\mathbf{x}_*'\beta_h^{(t)}, \tau_h^{(t)\,-1}). \tag{17}$$

Furthermore, this pool of densities provide also an estimate of the variance of the predictive density. One can leverage this information to estimate the uncertainty underlying the abstraction procedure in a specific state $\mathbf{x}_* \in \mathcal{S}$.

Sampling from the predictive density The same pool of T iterations can be used to sample from $f(\cdot|\mathbf{x}_*, \mathcal{D}, \alpha, \psi)$ in the following hierarchical way:

- randomly pick an iteration index $\hat{t} \in \{1, \dots, T\}$,
- given the parameters at iteration $\hat{t}$, we sample a component $\hat{h}^{(\hat{t})}$ from the discrete vector of weights $[\alpha, \bar{w}_{*,1}, \dots, \bar{w}_{*,k^{(\hat{t})}}]$, normalized by $(\alpha + w_*(\psi))$,
- from this $\hat{h}^{(\hat{t})}$ component, $\mathcal{N}(\cdot|\mathbf{x}_*'\beta_{\hat{h}^{(\hat{t})}}, \tau^{-1}_{\hat{h}^{(\hat{t})}})$ if $\hat{h}^{(\hat{t})} \neq 0$, we sample the desired value $\hat{y}$.

Iterating this procedure, we have an approximate simulation algorithm.

Consider a state of a d-dimensional system at time t, $\mathbf{x}_* = \eta_t$, and an abstract kernel trained on a given dataset and evaluated in $\mathbf{x}_*$. A sample from such kernel should return a full state of the system after a time Δt. In other words, the abstract kernel should approximate the joint probability distribution of the d variables composing the state space $\mathcal{S}$. Unfortunately, the proposed solution needs the response variable $y \in \mathcal{Y}$ to be one-dimensional, which means that we are actually approximating the marginal distribution of a single variable. Therefore, in order to simulate the full state of the system after a time Δt, i.e. $\eta_{t+\Delta t}$, we must sample from d different kernels, loosing the correlation between response variables. Nonetheless, the joint probability can be expressed in terms of chain of conditional probabilities. We are investigating a technique to approximate conditional distributions rather than marginals. The basic idea is to enlarge the input space by adding the response variables that condition the response of interest. Finally, the chain rule allows us to sample from the joint distribution while preserving the correlation between variables.

5 Experimental Results

We now validate the proposed Bayesian non-parametric approach on the two case studies introduced in Section 2: the SIR model and the Gene Regulatory Network (GRN) model.

Experimental setting Input data has been generated simulating the original CRN model by using both the direct and the τ-leaping SSA algorithms. The StochPy library (stochastic modeling in Python [12]) served for this purpose. The Gibbs sampling algorithm has been implemented in Python as well. All the computations were performed on a computer equipped with a CPU Intel x86 and 24 cores.

Setting the hyperparameters The proposed Bayesian method is nonparametric in the sense that it allows infinitely many parameters. However, some hyperparameters have to be fixed. As suggested in [6], we set, for both models, $\psi = n/25$, $a_\tau = 1$, $\beta_0 = \mathbf{0}$, $\Sigma_{\beta_0} = (\mathbf{X}'\mathbf{X})^{-1}$, $\nu_0 = d$, $V_0 = \mathbb{I}_d$, where d is the dimension of the state space, $a_0 = 1$ and $b_0 = 0.5$. In addition we set $\alpha = 0.5$ for SIR and, since we expect a larger number of cluster to be needed, we set $\alpha = 1$ for GRN. Data has been rescaled in order to have zero mean and variance one. This avoids sensitivity of the kernel function to different scales. Data has been scaled back after inference was performed, hence results are shown in the original scale. The Gibbs sampling algorithm performed $10,000$ iterations, with a burn-in period of $1,000$ iterations. The trace plots of different unknowns show that the convergence of the Gibbs sampler was rapid and mixing was good.

The training and validation set has been created as presented in Section 3. The time required to generate the dataset depends heavily on the length of the time step (Δt) considered and on the complexity of the model. For the SIR model, whose dynamic is rather simple, we fix $\Delta t = 0.1$ seconds. The time required to simulate $10,000$ trajectories starting from a given state is 125 second, using the direct SSA method, and 23 seconds, using the τ-leaping method. For the GRN model, in order to observe a strong multimodality, we choose a much larger time interval: $\Delta t = 400$ seconds. Here the computational effort required to generate the dataset is much higher. Generating $10,000$ trajectories takes around 40 hours with the direct SSA method, or around 20 hours with the τ-leaping approach. It is important to point out that the choice of Δt does not depend on the model itself, but rather on the intended use of abstraction. For instance, when using this abstraction in a multi-scale scenario (e.g. cells and tissues), Δt may be chosen as the integration step of the higher order model (e.g. the diffusion dynamics at the tissue level).

The training time of our approach increases with the number of training points. In order to sample from the predictive density and to compute the average predictive density we must fix the number T of Monte Carlo iterations that we are willing to consider. From a computational point of view, the sampling procedure is not affected by the dimension of T. However, computing the estimators for

inter-densities mean and variance takes longer as T increases. Leveraging the multi-core hardware available, we ran, for the GRN model, 12 parallel Gibbs sampling for each response variable and we mixed the final traces by taking the last 200 iterations of each process. For the simple SIR model, we ran just 2 parallel processes. By doing so, a greater mixing can be achieved. Therefore, for the GRN model, $T = 1200$, whereas, for the SIR model, $T = 400$.

5.1 SIR model

The constant parameters governing the transition rates are: $k_1 = 3$ and $k_2 = 1$. The model has been trained on a set of 500 observations. Experiments with more points showed similar results. Since we are considering a fixed population of size N, the model is two-dimensional ($d = 2$) and the number of recovered individuals is indeed $N - S - I$. We trained two separates models, one for each component/species. The first model predicts the number of susceptible individuals after Δt time units, whereas the second model predicts the number of infected individuals. The training of the two models has been performed in parallel and globally took 1.5 hours. The method automatically fix the number of cluster needed. The average number of clusters is 7.2 for the response variable S and 22.1 for the response variable I. In Figure 2, we can appreciate how a larger number of clusters is required as the intrinsic variance of the response I is larger with respect to the response variable S. Figure 2 shows the average density estimator, which is the average among T mixtures, against the empirical distribution obtained with $1,000$ SSA trajectories. The shaded area denotes the 95% confidence interval, indicating the variance among the T mixtures. Given $1,000$ samples from the true densities and $1,000$ samples from the approximate densities, the average histogram distances, over 10 validation points and 100 bins, are 0.362 for response variable S and 0.235 for the response variable I. The self-distance bound is 0.357. The average estimator is a faithful model of the true density for all the validation points. However, sometimes the inter-densities variances are large, which results in potentially high variability while sampling. In fact, as we sample from a single normal component of a randomly picked mixture, the specific component may deviate considerably from the average estimator. This fact may introduce instabilities in trajectories simulated for more Δt time instants, resulting in an artificial increase of variance. We are currently exploring alternative simulation strategies based on the average estimator to ameliorate this problem. In any case, sampling $10,000$ one-step trajectories from the abstract model takes on average 1.5 seconds, a considerable speed up of two orders of magnitude compared to the SSA algorithm.

5.2 GRN model

The constants governing the transition rates are: $k_{prodP} = 350$, $k_{prodM} = 300$, $k_{degM} = 0.001$, $k_{degP} = 1.5$, $k_{deact} = 166$, $k_{act} = 1$. In contrast with the SIR model, which is unimodal, we set a long time interval, $\Delta t = 400$ s, in order to observe strong multimodality and test our approach in such extreme scenario.

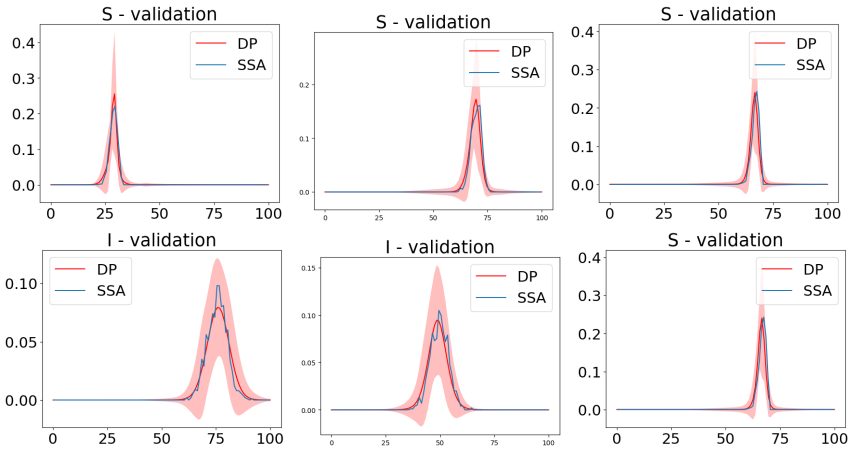

Fig. 2. SIR model: estimators of the average predictive densities and their confidence intervals, against the empirical simulated distributions for 3 validation points. Each column has a corresponding 2-dimensional value $\mathbf{x}_*$. The horizontal axis indicates the one-dimensional response space $\mathcal{Y}$. The first row shows the densities for the first component (susceptibles), while the second row shows results for the second component (infected). The grid on $\mathcal{Y}$, whose domain is $[0, 100] \subset \mathbb{N}$, contains 100 points. The empirical distribution is generated by $1,000$ SSA trajectories.

Since the species G^{ON} and G^{OFF} are constrained, i.e. $G^{ON} + G^{OFF} = 1$, the model is three-dimensional ($d = 3$). We analyze the performance of our approach in predicting the protein outcomes, since it is the species with the multi-stable behaviour. The model has been trained on a set of $2,000$ observations and training took around 20 hours. The average number of clusters is 154.5. Figure 3 shows the average density estimator and the 95% confidence interval against the empirical distribution obtained through SSA simulations.

The true and the approximate distributions are clearly distinguishable, but the main qualitative characteristics of the system are captured. The DP approach manages to recognize the 3 modes associated with highest probability. When it does not recognize a mode, it shows tails with high-variance. The last picture of Figure 3 shows the behaviour in situations where the new input point $\mathbf{x}_*$ has no neighbouring training points: it doesn't recognizes the modes but the variance is extremely large, reflecting the uncertainty in the reconstruction. Given $1,000$ samples from the true densities and 1000 samples from the approximate densities, the average histogram distances, over 25 validation points and 200 bins, is 0.12 (the self-distance bound is 0.504).

The analyzed case studies coincide with the ones presented in [3], thus, results may be compared. The use of deep neural networks renders the abstract model more accurate and the training is faster with respect to the non-parametric approach, but provides no estimate of the uncertainty. However, once the training is over, the time required by the two methods to simulate an abstract trajectory

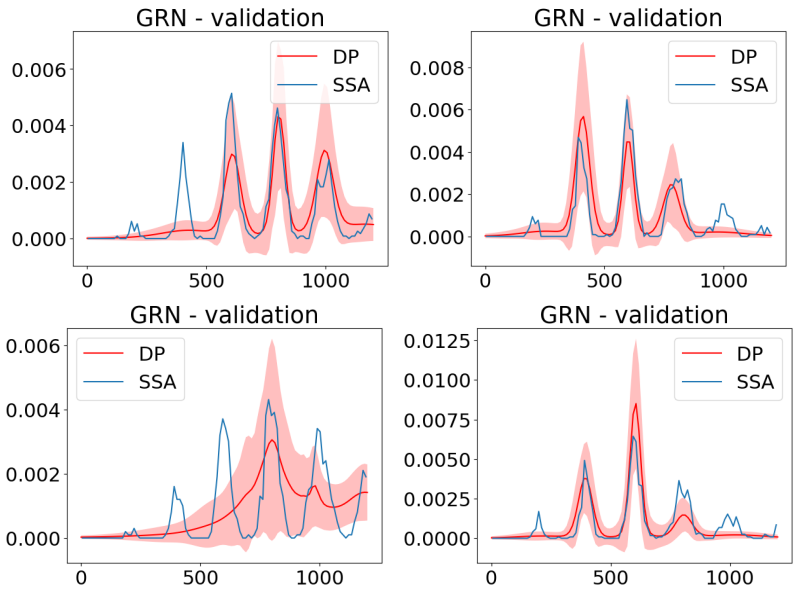

Fig. 3. GRN model: estimators of the average predictive densities and their confidence intervals, against the empirical simulated distributions for 4 validation points. The horizontal axis indicates the one-dimensional response space corresponding to the protein P. The grid on $\mathcal{Y}$ contains 100 points. The empirical distribution is generated by $1,000$ SSA trajectories.

is similar. It is reasonable to assume that the main reason behind the lower accuracy of our approach is due to the smaller training set, $2,000$ instead of $30,000$ data points. In fact, since the Monte Carlo method used is quite expensive, the Bayesian model is trained with relatively few points. The ability of the DP to recognize the modes is likely to increase as n grows. In order to significantly speed up the computations, we plan to develop a more efficient and compiled implementation of the algorithm. Finally, simulating $10,000$ one-step trajectories from the abstract model took on average 30 seconds, a tremendous speed-up compared to the SSA and τ-leaping algorithms.

6 Discussion

We presented a Bayesian approach to abstract the kernel of a Markov process. We used a mixture of Gaussian distributions with a mixing measure, a Dirichlet Process, flexible enough to vary accordingly with the state of the population. This work presents a first analysis on the performances of the proposed method, both in terms of accuracy and in terms of computational speed-up.

Results are encouraging, both in terms of the accuracy of the average estimator as a function of the input point and in terms of the computational gain. The

variance of the reconstructed density is also a good indicator of the uncertainty in the reconstruction, and can be used to guide an active learning step to refine the abstraction improving in target points with low accuracy.

Another important issue is to improve the accuracy of the method when it comes to iterate the sampling of the kernel to generate trajectories longer than Δt. One possibility is to sample from the average estimator, rather than from a single component of the mixture. The drawback in this case is the increased cost of simulation per step, which may be tamed by learning a simplified model (e.g. a mixture of Gaussians with a fixed number of components) that interpolates the average estimator.

An additional improvement, also in terms of speedup may come from using either variational approaches for density regression [15]. Finally, we also plan to extend this approach to approximate more general classes of models, like time inhomogeneous systems (including time as a covariate), including non-Markovian models.

References

1. Barber, D.: Bayesian reasoning and machine learning. Cambridge University Press (2012)
2. Bodei, C., Bortolussi, L., Chiarugi, D., Guerriero, M.L., Policriti, A., Romanel, A.: On the impact of discreteness and abstractions on modelling noise in gene regulatory networks. Computational Biology and Chemistry **56** (2015)
3. Bortolussi, L., Palmieri, L.: Deep abstractions of chemical reaction networks. In: Computational Methods in Systems Biology, CMSB 2018. pp. 21–38 (2018)
4. Cao, Y., Petzold, L.: Accuracy limitations and the measurement of errors in the stochastic simulation of chemically reacting systems. Journal of Computational Physics **212**(1), 6–24 (2006)
5. Deisboeck, T.S., Wang, Z., Macklin, P., Cristini, V.: Multiscale Cancer Modeling. Annual Review of Biomedical Engineering **13**(1), 127–155 (2011). https://doi.org/10.1146/annurev-bioeng-071910-124729
6. Dunson, D.B.: Empirical bayes density regression. Statistica Sinica **17**(2), 481 (2007)
7. Dunson, D.B., Pillai, N., Park, J.H.: Bayesian density regression. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **69**(2), 163–183 (2007)
8. Ferguson, T.S.: A bayesian analysis of some nonparametric problems. The annals of statistics pp. 209–230 (1973)
9. Gelman, A., Stern, H.S., Carlin, J.B., Dunson, D.B., Vehtari, A., Rubin, D.B.: Bayesian data analysis. Chapman and Hall/CRC (2013)
10. Gillespie, D.T., Petzold, L.: Numerical simulation for biochemical kinetics. Systems Modelling in Cellular Biology pp. 331–354 (2006)
11. Ishwaran, H., James, L.F.: Gibbs sampling methods for stick-breaking priors. Journal of the American Statistical Association **96**(453), 161–173 (2001)
12. Maarleveld, T.R., Olivier, B.G., Bruggeman, F.J.: Stochpy: a comprehensive, user-friendly tool for simulating stochastic biological processes. PloS one **8**(11), e79345 (2013)
13. Michaelides, M., Hillston, J., Sanguinetti, G.: Statistical abstraction for multi-scale spatio-temporal systems. In: Quantitative Evaluation of Systems, QEST 2017,. pp. 243–258 (2017)

14. Norris, J.R.: Markov chains. Cambridge University Press, Cambridge, UK; New York (1998)
15. Nott, D.J., Tan, S.L., Villani, M., Kohn, R.: Regression Density Estimation With Variational Methods and Stochastic Approximation. Journal of Computational and Graphical Statistics **21**(3), 797–820 (2012)
16. Palaniappan, S.K., Bertaux, F., Pichen, M., Fabre, E., Batt, G., Genest, B.: Abstracting the dynamics of biological pathways using information theory. Bioinformatics (2017)
17. Sethuraman, J.: A constructive definition of dirichlet priors. Statistica sinica pp. 639–650 (1994)