Active and Sparse Methods in Smoothed Model Checking

Active and sparse methods in smoothed model checking

Paul Piho, Jane Hillston

University of Edinburgh

Abstract. Smoothed model checking based on Gaussian process classification provides a powerful approach for statistical model checking of parametric continuous time Markov chain models. The method constructs a model for the functional dependence of satisfaction probability on the Markov chain parameters. This is done via Gaussian process inference methods from a limited number of observations for different parameter combinations. In this work we incorporate sparse variational methods and active learning into the smoothed model checking setting. We use these methods to improve the scalability of smoothed model checking. In particular, we see that active learning-based ideas for iteratively querying the simulation model for observations can be used to steer the model-checking to more informative areas of the parameter space and thus improve sample efficiency. We demonstrate that online extensions of sparse variational Gaussian process inference algorithms provide a scalable method for implementing active learning approaches for smoothed model checking.

1 Introduction

Stochastic modelling coupled with verification of logical properties via model checking has provided useful insights into the behaviour of the stochastic models from epidemiology, systems biology and networked computer systems. A large number of interesting models in these fields are too complex for the application of exact model checking methods [13]. To improve the scalability of model checking there has been significant work on statistical model checking that aims to estimate the satisfaction probability of logical properties based on independently sampled trajectories of a stochastic model [3].

This paper considers statistical model checking in the context of parametrised continuous time Markov chain models. Statistical model checking methods have generally considered single parametrisations of a model. Based on a large number of independent sample trajectories, one can estimate the probability of the model satisfying a specified logical property defined over individual sample trajectories. In order to gain insight across the entire parameter space associated with a model, it can be necessary to repeat the estimation procedures with different parametrisations to cover the whole space, which leads to poor scalability.

As an alternative, a model checking approach based on Gaussian process classification, named smoothed model checking, was proposed in [6]. The main

result of that paper was to show that under mild conditions the function which maps parameter values to satisfaction probabilities is smooth. Thus the problem can be solved as a Gaussian process classification problem where the aim is to estimate the function describing the satisfaction probability over the parameter space. Model checking results, returning a label true or false, of individual simulation trajectories are used as the training data to infer how the satisfaction probability depends on the model parameters. This method can be used to greatly reduce the number of simulation trajectories needed to estimate the satisfaction probability in exchange for some accuracy.

There are two aspects that limit the speed of such model checking procedures. Firstly, the computational cost of gathering the individual trajectories and secondly, the cost of (approximate) Gaussian process inference itself. Both benefit from keeping the number of gathered trajectories as low as possible while minimising the impact a smaller set of training data has on the accuracy of the methods. In order to keep the gathered sample size small we propose a method based on active learning. In particular, we make the observation that the parameter space of models is usually constrained to physically reasonable ranges. However, even when constrained to such ranges there can be large parts of the parameter space where the probability of satisfying a formula undergoes little change. Adaptively identifying parts of the parameter space where the satisfaction probability changes in order to decide where to concentrate the computational effort leads to improved algorithms for smoothed model checking.

The main contribution of this paper is considering smoothed model checking in a sparse online setting and proposing active learning strategies for querying the parameter space of a model. We make use of state of the art sparse variational Gaussian process inference methods and streaming variational inference with inducing points [8]. This combination of sparse inference methods and active learning improves the scalability of smoothed model checking in two ways. The use of sparse inference methods reduces the complexity of the underlying Gaussian process inference, while active learning query methods improve the sample efficiency by concentrating the model checking efforts to the parts of the parameter space where the satisfaction probability undergoes most change.

2 Related work

A wealth of literature exists on statistical model checking of stochastic systems. The use of statistical methods in the domain of formal verification is motivated by the fact that in order to perform statistical model checking it is only necessary to be able to simulate the model. Thus these methods can be used for systems where exact verification methods are infeasible including black-box systems [14]. In its classical formulation, this involves hypothesis testing [20] with respect to the desired (or undesired) property based on independent trials, or in this case, stochastic simulations.

In addition to the frequentist approaches based on hypothesis testing, there have been Bayesian approaches [12] to estimate the satisfaction probability of a

given logical formula. Our work follows the approach presented in [6] where the dependence of the satisfaction probability on model parameters is modelled as a Gaussian process classification problem.

The problem of deciding where to concentrate the model checking efforts is closely related to optimal experimental design. Experimental design problems are commonly treated as optimisation problems where the goal is to allocate resources in a way that allows the experimental goals to be reached more rapidly and thus with smaller costs [19]. This idea is also known in the machine learning literature as active learning [22]. The idea is to design learning algorithms that interactively query an oracle to label new data points.

In the context of model checking, active learning was used in [7] to solve a threshold synthesis problem which is closely related to the model checking problem considered in this paper. That approach used a base grid on the parameter space for initial estimation. The estimates were then refined around values where the satisfaction probability was close to a defined threshold. However, the threshold for synthesis has to be defined a priori making the introduced active step not applicable when we are interested in the satisfaction probability. We further address the scalability of the ideas presented by the authors of [7] by considering sparse approximation results for Gaussian process based model checking.

Finally, Bayesian optimisation is another example of active learning. Bayesian optimisation methods refine the posterior distribution over the black-box objective function based on function evaluations. An example relevant to model checking was given in [4] where Bayesian optimisation was used to optimise parameters of stochastic models to maximise robustness of the given logical specification.

3 Background

3.1 Continuous time Markov chains

Stochastic models are widely used to model a variety of phenomena in natural and engineered systems. We focus on a type of stochastic model commonly used in biological modelling, epidemiology and performance evaluation domains. Specifically, we consider continuous time Markov chain models (CTMCs). To define a CTMC we start by noting that it is a continuous-time stochastic process and thus defined as an indexed collection of random variables $\{\mathbf{X}\}_{t\in\mathbb{R}_{\geq 0}}$. We consider CTMCs defined over a finite state space S with an $|S|\times|S|$ matrix Q whose entries $q(i,j)$ satisfy

1. $0 \leq -q(i,i) < \infty$, 2. $0 \leq q(i,j)$ for $i \neq j$, 3. $\sum_j q(i,j) = 0$.

A CTMC is then defined by the following: for time indices $t_1 < t_2 \ldots < t_{n+1}$ and states $i_1, i_2, \ldots, i_{n+1}$ we have

$$\mathbb{P}(\mathbf{X}_{t_{n+1}} = i_{n+1} | \mathbf{X}_{t_n} = i_n, \ldots, \mathbf{X}_{t_1} = i_1) = p(i_{n+1}; t_{n+1} | i_n; t_n)$$

where $p(j;t|i;s)$ is the solution to the following Kolmogorov forward equation

$$\frac{\partial}{\partial t}p(j;t|i;s) = \sum_k p(k;t|i;s)q(k,j), \qquad \text{on } (s,\infty) \text{ with } p(j;s|i;s) = \delta_{ij}$$

with δ_{ij} being the Kronecker delta taking the value 1 if i and j are equal and the value 0 otherwise. By convention the sample trajectories of CTMCs are taken to be right-continuous.

In the rest of the paper we consider parametrised models $\mathcal{M}_{\mathbf{x}}$ and assume that the model $\mathcal{M}$ for a fixed parametrisation $\mathbf{x} \in \mathbb{R}^k$ defines a CTMC. Thus, the model $\mathcal{M}$ specifies a function mapping parameters $\mathbf{x}$ to generator matrix Q of the underlying CTMC. A commonly studied special class of CTMC models are population CTMCs where each state of the CTMC corresponds to a vector of counts. These counts are used to model the aggregate counts of groups of indistinguishable agents in a system. In biological modelling and epidemiology such models are often defined as chemical reaction networks (CRN).

Example 1. Let us consider the following susceptible-infected-recovered (SIR) model defined as a CRN

$$S + I \xrightarrow{k_I} I + I \qquad\qquad I \xrightarrow{k_R} R$$

where S gives the number of susceptible, I the infected and R the recovered individuals in the system. The first type of transition corresponds to infected and susceptible individuals interacting, resulting in the number of infected individuals increasing and the number of susceptible decreasing. The second type of transition corresponds to recovery of an infected individual and results in the number of infected decreasing and the number of recovered increasing. The states of the underlying CTMC keep track of the counts of different individuals in the system. For the example let us set the initial conditions to $(95, 5, 0)$ — at time 0 there are 95 susceptible, 5 infected and 0 recovered individuals in the system. The parameters k_I and k_R give the infection and recovery rates respectively. We revisit this example throughout the paper to illustrate the presented concepts.

3.2 Smoothed model checking

Smoothed model checking was introduced in [6] as a scalable method for statistical model checking where Gaussian process classification methods were used to infer the functional dependence between a parametrisation of a model and the satisfaction probability given a logical specification.

As described in Section 3.1, suppose we have a model $\mathcal{M}_{\mathbf{x}}$ parametrised by vector of values $\mathbf{x} \in \mathbb{R}^d$ such that the model $\mathcal{M}$ for a fixed parametrisation $\mathbf{x}$ defines a CTMC. Additionally assume we have a logical property φ we want to check against. The logical properties we consider here are defined as a mapping from the time trajectories over the states of $\mathcal{M}_{\mathbf{x}}$ to $\{0, 1\}$ corresponding to whether the property holds for a given sample trajectory of $\mathcal{M}_{\mathbf{x}}$ or not. One way to define such mappings would be, for example, to specify the properties in metric

interval temporal logic (MiTL) [15] or signal temporal logic (STL) [10] and map the paths satisfying the properties to 1 and those not satisfying the properties to 0. Through sampling multiple trajectories for the same parametrisation we gain an estimate of the satisfaction probability corresponding to the parametrisation.

With that in mind, a logical property φ with respect to $\mathcal{M}_{\mathbf{x}}$ can be seen to give rise to a Bernoulli random variable. The binary outcomes of the random variable correspond to whether or not a randomly sampled trajectory of $\mathcal{M}_{\mathbf{x}}$ satisfies the property φ. We introduce the notation $f_\varphi(\mathbf{x})$ for the parameter of the said Bernoulli random variable given the model parameters $\mathbf{x}$. In particular, samples from the distribution $Bernoulli(f_\varphi(\mathbf{x}))$ model whether a randomly sampled trajectory of $\mathcal{M}_{\mathbf{x}}$ satisfies φ — for a parameter value $\mathbf{x}$ the logical property is said to be satisfied with probability $f_\varphi(\mathbf{x})$.

A naive approach for estimating $f_\varphi(\mathbf{x})$ at a given parametrisation $\mathbf{x}$ is to gather a large number N of sample trajectories and give simple Monte Carlo estimates for the $f_\varphi(\mathbf{x})$ by dividing the number of trajectories where the property holds by the total number of sampled trajectories N. An accurate estimate requires a large number of samples. However, having such an estimate at a set of given parametrisations does not provide us with a rigorous way to estimate the satisfaction function at a nearby point.

In [6] the authors considered population CTMCs. It was shown that the introduced satisfaction probability $f_\varphi(\mathbf{x})$ is a smooth function of $\mathbf{x}$ under the following conditions: the transition rates of the CTMC $\mathcal{M}_{\mathbf{x}}$ depend smoothly on the parameters $\mathbf{x}$; and the transition rates depend polynomially on the state vector $\mathbf{X}$ of the CTMC.

The result was exploited by treating the estimation of the satisfaction function $f_\varphi(\mathbf{x})$ as a Gaussian process classification problem. The main benefit of this approach is that, based on sampled model checking results, we can reconstruct an approximation for the functional dependence between the parameters and satisfaction probability. This makes it easy to make predictions about the satisfaction probability at previously unseen parametrisations.

Simulating $\mathcal{M}_{\mathbf{x}}$ we gather a finite set of observations $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1, \cdots, n\}$ where $\mathbf{x}_i$ are the parametrisations of the model and y_i correspond to model checking output over single trajectories. For classification problems, a Gaussian process prior with mean m and kernel k is placed over a latent function

$$g_\varphi(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))\,.$$

Here, let us consider the standard squared exponential kernel defined by

$$k(\mathbf{x}, \mathbf{x}') = a^2 \exp\left(-\frac{|\mathbf{x} - \mathbf{x}'|^2}{2l}\right)$$

where a^2 is the amplitude and ℓ is the length scale parameter governing how far two distinct points have to be in order to be considered uncorrelated.

The function g_φ is then squashed through the standard logistic or probit transformation σ so that the composition $\sigma(g_\varphi(\mathbf{x}))$ takes values between 0 and 1. The quantity $\sigma(g_\varphi(\mathbf{x}))$ is interpreted as the probability that φ holds given model

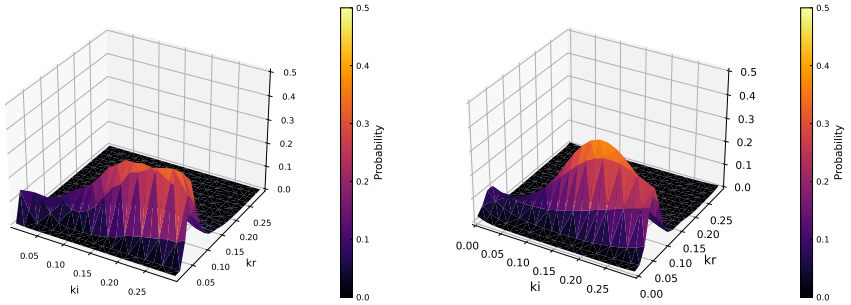


Fig. 1: Left is the baseline satisfaction probability surface. Satisfaction probability estimated for parameters on a regular 20×20. At each parameter the estimate is based on 3000 SSA trajectories. Right is the smoothed model checking satisfaction probability surface. Training data is constructed from a set of 10 SSA trajectories at parameter points on a regular 15×15 grid.

parametrisation $\mathbf{x}$ and thus estimates the probability $f_\varphi(\mathbf{x})$ that a simulation trajectory for parameters $\mathbf{x}$ satisfies the property φ.

The general aim of Gaussian process inference is to find the distribution $p(g_\varphi(\mathbf{x}^*)|\mathcal{D})$ over the values g_φ at some test point $\mathbf{x}^*$ given the set of training observations $\mathcal{D}$. This distribution is then used to produce a probabilistic prediction at parameter $\mathbf{x}^*$ of $\sigma(g_\varphi(\mathbf{x}^*)) \approx f_\varphi(\mathbf{x}^*)$. We present details of inference in the next section. This section is ended by returning to the running SIR example.

Example 2. The property we consider is the following: there always exists an infected agent in the population in the time interval $(0.0, 100.0)$ and in the time interval $(100.0, 120.0)$ the number of infected becomes 0. Constraining the parameters to the ranges $k_I \in [0.005, 0.3]$ and $k_R \in [0.005, 0.3]$ gives satisfaction probabilities as depicted in Figure 1. There each estimate on the 20×20 grid is calculated based on 3000 stochastic simulation algorithm (SSA) sample runs of the model. For comparison, Figure 1 also gives the results of the smoothed model checking where 10 sample trajectories are drawn for each parameter on the 12×12 grid. The smoothed model checking approximation for the model checking problem shows good agreement with the baseline surface and is much faster to perform.

3.3 Variational inference with inducing points

In order to infer the latent Gaussian process g_φ based on training data $\mathcal{D}$ we have to deal with two problems. Firstly the inference is analytically intractable due to the non-Gaussian likelihood model provided by Bernoulli observations. To counter this there exists a wealth of approximate inference schemes like Laplace approximation, expectation propagation [16, 11], and variational inference methods [23]. Here we consider variational inference. The second problem is that the

methods for inference in Gaussian process models have cubic complexity in the number of training cases. To address that, there exist sparse approximations based on inducing variables. Sparse variational methods [9, 23] are popular methods for reducing the complexity of Gaussian process inference by constructing an approximation based on a small set of inducing points that are typically selected from training data. In this section we detail the inference procedure.

Variational inference methods choose a parametric class of variational distributions for the posterior and minimising the KL-divergence between the real posterior and the approximate posterior. To accommodate large training data sets we work with sparse variational methods. We start by defining the prior distribution

$$p(\mathbf{g}_\varphi, \mathbf{u}) = \mathcal{N}\left(\begin{bmatrix}\mathbf{g}_\varphi \\ \mathbf{u}\end{bmatrix} \middle| \mathbf{0}, \begin{bmatrix}\mathbf{K}_{nn} & \mathbf{K}_{nm} \\ \mathbf{K}_{nm}^T & \mathbf{K}_{mm}\end{bmatrix}\right)$$

where $\mathbf{g}_\varphi$ is a vector of n latent function values $[g_\varphi(\mathbf{x}_1), \cdots, g_\varphi(\mathbf{x}_n)]$. Similarly, $\mathbf{u}$ is a vector of m latent function values $[g_\varphi(\mathbf{z}_1), \cdots, g_\varphi(\mathbf{z}_m)]$ evaluated at chosen *inducing points* $\mathbf{z}_i$. The matrices $\mathbf{K}_{nn}$, $\mathbf{K}_{nm}$ and $\mathbf{K}_{mm}$ are defined by the kernel function. In particular, the (i, j)-th element of the matrix $\mathbf{K}_{nn}$ is given by $k(\mathbf{x}_i, \mathbf{x}_j)$. Similarly, $\mathbf{K}_{nm}$ gives the kernel matrix between the training points $\mathbf{x}$ and the inducing points $\mathbf{z}$ and $\mathbf{K}_{mm}$ gives the kernel matrix between the locations of inducing points. We then fit the variational posterior at those points rather than the whole set of training data points. The assumption we are making is that $p(g_\varphi(\mathbf{x}_*)|\mathbf{g}_\varphi, \mathbf{u}) = p(g_\varphi(\mathbf{x}_*)|\mathbf{u})$. That is, the inducing values $\mathbf{u}$ are a sufficient statistic for a function value at a test point $\mathbf{x}_*$.

Under this assumption we make predictions at a test point $\mathbf{x}^*$ as follows:

$$p(g_\varphi(\mathbf{x}_*)|\mathbf{y}) = \int p(g_\varphi(\mathbf{x}_*), \mathbf{u}|\mathbf{y}) d\mathbf{u} = \int p(g_\varphi(\mathbf{x}_*)|\mathbf{u}) p(\mathbf{u}|\mathbf{y}) d\mathbf{u}\,.$$

Thus, we need posterior distribution $p(\mathbf{u}|\mathbf{y})$ at the inducing points. Here, as mentioned, we consider variational approximations where $p(\mathbf{u}|\mathbf{y})$ is approximated by a multivariate Gaussian $q(\mathbf{u})$ making the expression for $p(g_\varphi(\mathbf{x}_*)|\mathbf{y})$ tractable. Finding the parameters of $q(\mathbf{u})$ is done by minimising the KL divergence between the approximate posterior $q(\mathbf{u})$ and true posterior $p(\mathbf{u}|\mathbf{y})$. In particular, we have

$$\mathcal{D}_{KL}(q(\mathbf{u}), p(\mathbf{u}|\mathbf{y})) = \int q(\mathbf{u}) \log \frac{q(\mathbf{u})}{p(\mathbf{u}|\mathbf{y})} d\mathbf{u} = -\big\langle \log \frac{p(\mathbf{y}, \mathbf{u})}{q(\mathbf{u})} \big\rangle_{q(\mathbf{u})} + \log p(\mathbf{y}) \quad (1)$$

where $\langle \cdot \rangle_{q(\mathbf{u})}$ denotes the expectation with respect to distribution $q(\mathbf{u})$. The term $\log p(\mathbf{y})$ is known as the log marginal likelihood. In the following we use the well-known Jensen's inequality[1] to derive a lower bound for the log marginal likelihood. As log function is concave we get the following:

$$\begin{aligned} \log p(\mathbf{y}) &= \log \int p(\mathbf{y}, \mathbf{u}) d\mathbf{u} = \log \Big\langle \frac{p(\mathbf{y}, \mathbf{u})}{q(\mathbf{u})} \Big\rangle_{q(\mathbf{u})} \geq \Big\langle \log \frac{p(\mathbf{y}, \mathbf{u})}{q(\mathbf{u})} \Big\rangle_{q(\mathbf{u})} \\ &= \langle \log p(\mathbf{y}|\mathbf{u}) \rangle_{q(\mathbf{u})} - \mathcal{D}_{KL}(q(\mathbf{u}), p(\mathbf{u}))\,. \end{aligned} \quad (2)$$

[1] For a concave function f and a random variable X we have the following well-known inequality: $f\langle X \rangle \geq \langle f(X) \rangle$.

The right-hand side of the inequality 2 is known as the evidence-based lower bound or ELBO. Now note that the first term in the expression for KL-divergence in Equation 1 is exactly the derived ELBO. As the KL-divergence is always non-negative and ELBO serves as a lower bound for the log marginal likelihood $p(\mathbf{y})$, then maximising the ELBO minimises the KL-divergence between the approximate and true posteriors $q(\mathbf{u})$ and $p(\mathbf{u}|\mathbf{y})$.

Choosing our approximating family of variational distribution to be multivariate Gaussian makes the KL term in ELBO easy to evaluate. The integral in the expectation term $\langle \log p(\mathbf{y}|\mathbf{u}) \rangle_{q(\mathbf{u})}$ can be computed via numerical approximation schemes making it possible to use ELBO as a utility function for optimising the parameters of the approximate posterior $q(\mathbf{u})$ via gradient ascent. When $q(\mathbf{u})$ is chosen to be a multivariate Gaussian these parameters are the mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. With the approximation $p(\mathbf{u}|\mathbf{y}) \approx q(\mathbf{u})$ the predictions are given by the integral

$$p(g_\varphi(\mathbf{x}_*)|\mathbf{y}) \approx \int p(g_\varphi(\mathbf{x}_*)|\mathbf{u}) q(\mathbf{u}) d\mathbf{u}\,.$$

This can be shown [18] to be a probability density function of a Normal distribution with the following mean μ and variance σ^2

$$\begin{aligned}
\mu_* &= k(\mathbf{x}_*, \mathbf{u}) K_{mm}^{-1} \boldsymbol{\mu} \\
\sigma_*^2 &= k(\mathbf{x}_*, \mathbf{x}_*) - k(\mathbf{x}_*, \mathbf{u}) K_{mm}^{-1} \left[K_{mm} - \boldsymbol{\Sigma} \right]^{-1} \left[k(\mathbf{x}_*, \mathbf{u}) K_{mm}^{-1} \right]^T .
\end{aligned}$$

Note that the terms in ELBO depend on the chosen kernel and in particular the kernel hyperparameters. As mentioned, the ELBO is maximised directly via gradient ascent with respect to the parameters of the variational distribution. The kernel hyperparameter can be tuned in the same fashion. A common approximate technique we use in this paper is to interleave the optimisation steps in the variational distribution parameters with optimisation steps in the hyperparameters.

We have made the assumption that the posterior distribution is fitted at a selection of inducing points $\mathbf{z}$ such that number of inducing points is much smaller than the whole training data set. There exists a variety of possible methods to select the inducing points. For simplicity in this paper we use a regular grid over the parameter space as our inducing points.

3.4 Active learning

Active learning methods in machine learning are a family of methods which may query data instances to be labelled for training by an *oracle* [21]. The fundamental question asked by active learning research is whether or not these methods can achieve higher accuracy than passive methods with fewer labelled examples. This is closely related to the established area of optimal experimental design, where the goal is to allocate experimental resources in a way that reduces uncertainty about a quantity or function of interest [19, 22].

In the case of Gaussian process classification problems like smoothed model checking, an active learning procedure can be set up as follows. An *active learner* consists of a classifier learning algorithm $\mathcal{A}$ and a *query function* $\mathcal{Q}$. The query function is used to select an unlabelled sample u from the pool of unlabelled samples $\mathcal{U}$. This sample is then labelled by an oracle. In the case of stochastic model checking, the pool of unlabelled samples $\mathcal{U}$ corresponds to a subset of the possible parametrisations for the model. An oracle is implemented by running the stochastic simulation for the selected parametrisation and then model checking the resulting trajectory.

The above describes a pool-based active learner. Common formulations of such pool-based learners select a single unlabelled sample at each iteration to be sampled. However, in many applications it is more natural to acquire labels for multiple training instances at once. In particular, the query function $\mathcal{Q}$ selects a subset $U \subset \mathcal{U}$. We see in the next section that the sparse inference methods can be extended to a setting where batches of training data become available over time making it natural to decide on a query function that selects batches of queries. The main difficulty of selecting a batch of queries instead of a single query is that the instances in the subset U need to be both informative and diverse in order to make the best use of the available labelling resources.

4 Active model checking

The shape and properties of the functional dependence of satisfaction for a logical specification with respect to parameters are generally not known a priori and can exhibit a variety of properties. For example, in the running example much of the sampling was performed in completely flat regions of the parameter space. Thus the key challenge addressed in this section is where to sample to make the posterior estimates as informative as possible about the underlying mechanics. We aim to decide on the regions where the satisfaction probability surface is not flat and concentrate most of our model checking effort there. To that end we introduce the main contribution of this paper — active sparse model checking.

The general outline of the procedure is given by Algorithm 1. The first step, given by the procedure *generate_initial_data*, is to simulate the initial data set $\mathcal{D}_{old}$ via stochastic simulation of the CTMC model $\mathcal{M}$ for a sample of the parameter space $\mathcal{X}$ and checking whether or not the individual trajectories satisfy the property φ or not. The initial set of parameter samples can, for example, be a regular grid or sampled uniformly from the parameter space.

In general the inducing points are then chosen based on the results $\mathcal{D}_{old}$ and adjusted as new data is seen. However, for simplicity we are going to set the inducing points so that they form a regular grid over the parameter space and keep them fixed throughout the active iterations. The posterior at inducing points $\mathbf{z_v}$ is then initialised as a multivariate Gaussian $q(\mathbf{v}) = \mathcal{N}(\mathbf{0}, \mathbf{I}))$ with 0 mean and identity covariance matrix. Each iteration of the model checking loop will first update the variational posterior $q(\mathbf{v})$ via *update_variational*. The details of how this is done in a sparse streaming setting are given in Section 4.1. Secondly,

Algorithm 1 Active smoothed model checking

procedure MODEL CHECKING(model $\mathcal{M}$, property φ, parameter space $\mathcal{X}$)
  $\mathcal{D}_{old} \leftarrow$ *generate_initial_data*$(\mathcal{M}, \varphi, \mathcal{X})$
  $\mathcal{D}_{new} \leftarrow \mathcal{D}_{old}$
  $\mathbf{z_u} \leftarrow$ *inducing_points*$(\mathcal{D}_{old})$
  $\mathbf{z_v} \leftarrow \mathbf{z_u}$
  $q(\mathbf{v}) \leftarrow$ *initialise_posterior*$(\mathbf{v})$
  while *true* **do**
    $q(\mathbf{v}) \leftarrow$ *update_variational*$(q(\mathbf{u}), \mathbf{v}, \mathcal{D}_{new}, \mathcal{D}_{old})$
    $\mathcal{D}_{old} \leftarrow \mathcal{D}_{old} \cup \mathcal{D}_{new}$
    $\mathcal{D}_{new} \leftarrow$ *query_new*$(q(\mathbf{v}), \mathcal{M}, \varphi, \mathcal{X})$
  end while
  return $q(\mathbf{v})$
end procedure

each iteration uses the fitted approximate posterior to query new points in the parameter space to perform model checking through the *query_new* procedure. The proposed query functions are discussed in Section 4.2.

There are two issues to be resolved before the procedure can be implemented. First is that the direct use of ELBO as introduced in Section 3.3 does not suffice in the online setting where new data becomes available in batches. Second is the challenge of choosing an appropriate query function that is going to suggest more points in the parameter space at which to gather more model checking data. These will be addressed in the following sections.

4.1 Streaming setting

In order to incorporate active learning ideas into the Gaussian process based model checking approach we need to address the problem that not all of the training data is available a priori. For our purposes it is important to be able to conduct inference in a streaming setting where data is gradually added to the model. A naive approach would refit a Gaussian process from scratch every time a new batch of data arrives. However, with potentially large data sets this becomes infeasible. To perform sparse variational inference in a scalable way the method needs to avoid revisiting previously considered data points. In particular, we consider the method proposed in [8] that derives a correction to ELBO that allows us to incorporate streaming data incrementally into the posterior estimate.

The main question is how to update the variational approximation to the posterior at time step n, denoted $q_{old}(\mathbf{u})$, to form an approximation at the time step $n+1$, denoted $q_{new}(\mathbf{v})$. In the following we note the variational posteriors q_{old} and q_{new} at $\mathbf{g}_\varphi$ and inducing values $\mathbf{u}$ and $\mathbf{v}$, respectively, are approximations to the true posteriors given observations $\mathbf{y}_{old}$ and $\mathbf{y}_{new}$. It was shown in [8] that

the lower bound of $\log p(\mathbf{y}_{new}|\mathbf{y}_{old})$ becomes

$$\int q_{new}(\mathbf{g}_\varphi, \mathbf{v}) \log p(\mathbf{y}_{new}|\mathbf{g}_\varphi, \mathbf{v}) d(\mathbf{g}_\varphi, \mathbf{v}) - \mathcal{D}_{KL}(q_{new}(\mathbf{v}), p(\mathbf{v})) \\ -\mathcal{D}_{KL}(q_{new}(\mathbf{u}), q_{old}(\mathbf{u})) + \mathcal{D}_{KL}(q_{new}(\mathbf{u}), p(\mathbf{u}))\,.$$

The above can be interpreted as follows: the first two terms give the ELBO under the assumption that the new data seen at iteration $n+1$ is the whole data set; the final two terms take into account the old likelihood through the approximate posteriors at old inducing points and the prior $p(\mathbf{u})$. This allows us to implement an online version of the smoothed model checking where the observation data arrives in batches.

4.2 Query strategies

As discussed in Section 3.4, in order to implement an active learning method for model checking we need to decide which new parameters are tested based on the existing information. In the following we consider two query strategies for active model checking.

Predictive variance The first approach is a commonly used experimental design strategy which aims to minimise the predictive variance. Recall that in smoothed model checking for a property φ we fit a latent Gaussian process g_φ. The posterior satisfaction probability for parameter $\mathbf{x}_*$ given the GP g_φ is then calculated via

$$p(y_* = 1|\mathcal{D}, \mathbf{x}_*) = \int \sigma(g_\varphi(\mathbf{x}_*)) p(g_\varphi(\mathbf{x}_*)|\mathcal{D}) dg_\varphi(\mathbf{x}_*)\,.$$

The above can also be seen as the expectation of $\sigma(g_\varphi(\mathbf{x}_*)$ with respect to the distribution $g_\varphi(\mathbf{x}_*)$, denoted $\mathbb{E}\left[\sigma(g_\varphi(\mathbf{x}_*)\right]$. Similarly, we can consider the variance of this estimate

$$\mathbb{E}\left[\sigma(g_\varphi(\mathbf{x}_*)^2\right] - \mathbb{E}\left[\sigma(g_\varphi(\mathbf{x}_*)\right]^2\,.$$

Our aim is then to iteratively train the Gaussian process model so that predictive variance over the parameter space is minimised.

Before giving the outline of the proposed procedure we address the issue of redundancy in the query points. As pointed out in Section 3.4, simply taking a set of points with the highest predictive variance leads to querying parameters that are clustered together. We can overcome this problem by clustering the pool of unlabelled samples $\mathcal{U}$ from which the query choice is made. In particular, the points with the highest predictive variance are chosen from a pool of samples where the redundancy is already reduced. Informally, this leads to the following basic outline of the procedure:

1. Sample an initial set of training points or parametrisations $\mathbf{x}$ of the model (via uniform or Latin hypercube sampling or taking points on a regular grid) and conduct model checking based on sampled trajectories. These points are used to fit the first iteration of the Gaussian process model.

2. For the next iteration we randomly sample another set of points U and cluster them via regular kmeans clustering algorithm. From the set of cluster centres U_k the query function $\mathcal{Q}$ selects a set of points for model checking. The query function is simply defined by taking the subset U^* of cluster centres U_k where the predictive variance, as defined above, is the highest. This concentrates the sampling to points where the model is most uncertain about its prediction.
3. The points in U^* are labelled by simulating the model for the parametrisations in U^* and checking the resulting trajectories against the logic specification φ. The results are incorporated into the Gaussian process model via the streaming method discussed in Section 4.1.
4. Repeat points 2 and 3 until a set computational budget is exhausted.

Predictive gradient The second strategy we consider is based on the predictive mean $\bar{g}_\varphi(\mathbf{x})$ of the Gaussian process. Our aim is to concentrate the sampling at the locations where the predictive mean undergoes the most rapid change. This requires gradients of the predictive mean.

We recall from Section 3.3 that for a variational posterior $q(\mathbf{u})$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, the posterior mean at a point $\mathbf{x}$ is given by

$$\bar{g}_\varphi(\mathbf{x}) = k(\mathbf{x}, \mathbf{z_u}) K_{mm}^{-1} \boldsymbol{\mu} \stackrel{\text{def}}{=} k(\mathbf{x}, \mathbf{z_u}) \boldsymbol{\alpha} \,.$$

Only the first part, the kernel function, depends on $\mathbf{x}$. Thus, in order to get the derivative of the predictive mean we need to differentiate $k(\mathbf{x}, \mathbf{z_u})$. Recall that in this paper we chose to work with the squared exponential kernel given by

$$k(\mathbf{x}, \mathbf{z}_{\mathbf{u}_i}) = \exp\left(-\frac{|\mathbf{x} - \mathbf{z}_{\mathbf{u}_i}|^2}{2\ell}\right) .$$

We have used $\mathbf{z}_{\mathbf{u}_i}$ to denote a single inducing point in the set of inducing points $\mathbf{z_u}$. Thus, the derivative of $k(\mathbf{x}, \mathbf{z}_{\mathbf{u}_i})$ with respect to $\mathbf{x}$ is given by

$$\frac{dk(\mathbf{x}, \mathbf{z}_{\mathbf{u}_i})}{d\mathbf{x}} = -\frac{\mathbf{x} - \mathbf{z}_{\mathbf{u}_i}}{\ell} \exp\left(-\frac{|\mathbf{x} - \mathbf{z}_{\mathbf{u}_i}|^2}{2\ell}\right) = -\frac{\mathbf{x} - \mathbf{z}_{\mathbf{u}_i}}{\ell} k(\mathbf{x}, \mathbf{z}_{\mathbf{u}_i}) \,. \tag{3}$$

Equation 3 is given for a single inducing point $\mathbf{z}_{\mathbf{u}_i}$. In order to compute the derivative of the posterior mean we need to concatenate this derivative for all m inducing points. Thus, we get

$$\frac{d\bar{g}_\varphi(\mathbf{x})}{d\mathbf{x}} = -\ell^{-1} \begin{bmatrix} \mathbf{x} - \mathbf{z}_{\mathbf{u}_1} \\ \cdots \\ \mathbf{x} - \mathbf{z}_{\mathbf{u}_m} \end{bmatrix} (k(\mathbf{x}, \mathbf{z_u}) \odot \boldsymbol{\alpha})$$

where $\odot$ denotes element-wise multiplication. Given this we can proceed as in the case of the predictive variance. The only change is that instead of considering the predictive variance for each sampled set of parameters we calculate the norm of $\frac{d\bar{g}_\varphi(\mathbf{x})}{d\mathbf{x}}$ and define the query function to choose a subset U^* of cluster centres U_k with the highest norms.

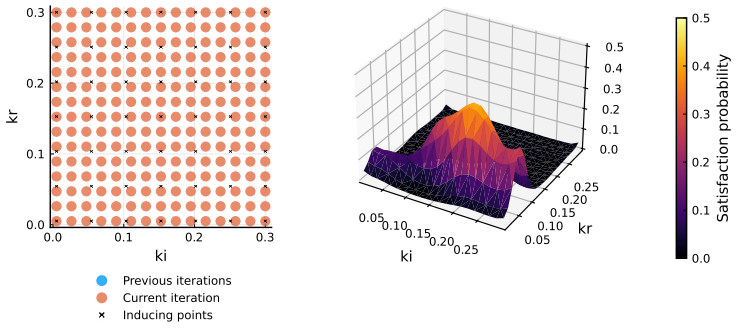


Fig. 2: Mean satisfaction probability surface for sparse smoothed model checking. Inducing points are set as a regular 7×7 grid. Training data is constructed from a set of 10 SSA trajectories at parameter points on a 15×15 grid.

4.3 Implementation

The prototype implementation in written in the Julia programming language and makes use of the tools provided as part of Julia Gaussian Processes repositories [1] to set up the Gaussian process models. The variational inference-based fitting which maximises the ELBO with respect to parameters of the posterior distributions as well as the kernel hyperparameters was implemented for all of the sparse methods. For the standard smoothed model checking we use the U-Check tool [5] available on GitHub [2]. The CTMC models are defined as CRNs with tools provided as part of the SciML ecosystem for scientific simulations [17]. The simulations were carried out on a laptop with Intel i7-10750H CPU.

4.4 Results

In this section we evaluate the proposed active learning methods for model checking on the running SIR example. The methods are compared to the baseline naive stochastic simulation-based model checking and smoothed model checking without the sparse approximation and the active step. We present several metrics for comparing the smoothed model checking results with the empirical mean based on stochastic simulation. The first is the mean and standard deviation of the difference between the mean probability predicted by the fitted Gaussian processes and the empirical mean from the stochastic simulation results at each of the points on the 20×20 grid. Secondly, we consider the maximum difference between the predicted mean probability and the naive empirical mean. Finally, we give the root-mean-square error (RMSE) $\sqrt{\frac{1}{N}\sum\left(\bar{g}_\varphi(\mathbf{x}_i) - \bar{f}_\varphi(\mathbf{x}_i)\right)^2}$ where $\bar{g}_\varphi(\mathbf{x}_i)$ is the predicted mean satisfaction probability for parametrisation $\mathbf{x}_i$. We denote by $\bar{f}_\varphi(\mathbf{x}_i)$ the empirical estimate of the satisfaction probability at $\mathbf{x}_i$ given 3000 sample trajectories.

In the active learning experiments we start with a 12×12 grid followed by an active iteration where an additional 81 parameter points are chosen to refine the

Table 1: Comparison of accuracy for smoothed model checking and the sparse and active learning extensions. All of the smoothed model checking methods are assigned the same computational budget of 225 parameter values with training data constructed from 10 SSA trajectories of the SIR model at each parametrisation. For smoothed MC and sparse smoothed MC the parameters are considered on a regular grid.

Method	Error mean/var	Maximum	RMSE
Smoothed MC (U-Check)	$(0.044, 0.042)$	0.166	0.666
Sparse smoothed MC	$(0.042, 0.036)$	0.147	0.6
Active sparse smoothed MC			
Predictive variance	$(0.033, 0.029)$	**0.131**	0.479
Predictive gradient	$(\mathbf{0.03}, \mathbf{0.026})$	0.14	**0.436**
Random sampling	$(0.049, 0.039)$	0.149	0.681

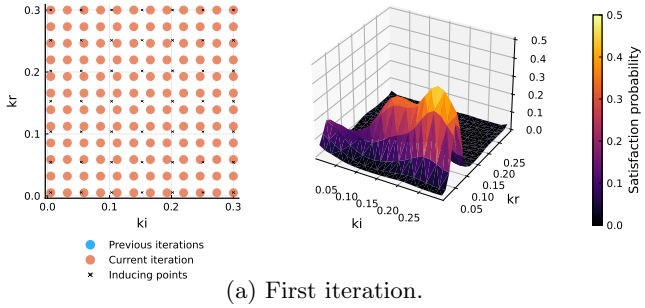
Table 2: Comparison of computation times (in seconds) for smoothed model checking and the sparse and active learning extensions. The hyperparameter tuning time is included in the inference column.

Method	SSA	Inference	Active query	Total
Naive statistical MC	191.5	N/A	N/A	191.5
Smoothed MC (U-Check tool)	0.21	16.8	N/A	17.0
Sparse smoothed MC	0.48	4.1	N/A	4.5
Active sparse smoothed MC				
Predictive variance	0.57	5.7	0.003	6.2
Predictive gradient	0.55	3.1	0.2	3.9
Random sampling	0.60	2.9	0.00	3.5

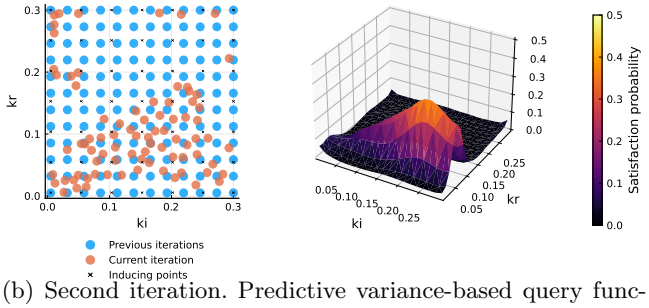
approximation for a total of 225 training points. At each parameter point the model checking is conducted for 10 sample trajectories. The inducing points are initialised by choosing a regular 7×7 grid and kept constant for the remainder of the fitting procedure. Similarly we present the results for sparse smoothed model checking for a 15×15 grid with the 7×7 grid of inducing points, as well as smoothed model checking where inducing points are not chosen.

Figure 2 gives the mean satisfaction probability surface based on the fitted sparse Gaussian process. Figure 3 presents the evolution of the predictive mean surface through two active learning iterations. All figures are accompanied by the scatter plots showing where the samples were drawn.

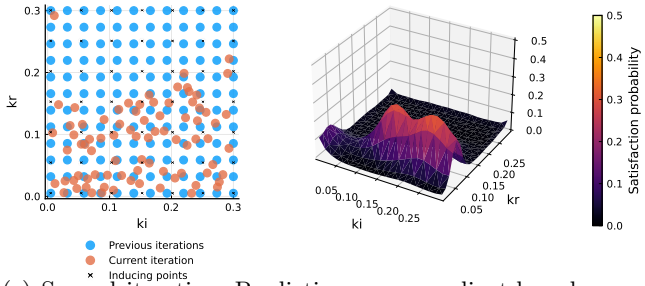
The results for accuracy are summarised in Table 1. Table 1 gives the comparisons for each point on the 20×20 grid where the naive model checking was conducted and satisfaction probability estimates exceeding 0.02. This is done to concentrate the analysis to the parts of the parameter space where the surface is not completely flat. As expected, the main benefit of the sparse methods comes from significant reductions in computation costs. Surprisingly however, the stan-



(a) First iteration.



(b) Second iteration. Predictive variance-based query function.



(c) Second iteration. Predictive mean gradient-based query function.

Fig. 3: Mean satisfaction probability surfaces for the active and sparse smoothed model checking methods with 2 iterations. The first iteration of the active learning-based methods fits the Gaussian process inference model based on 10 model checking results for each parameter on a regular 12×12 grid. The active step is then used to exhaust the total computational budget of 225 parameters and refine the approximation.

dard sparse method without active step performs better than the method provided by the U-Check tool with respect to the error metrics considered. The reasons for this may be the differences in the recovered hyperparameters — no-

tably, amplitude ≈ 2.7 for U-Check versus ≈ 4.5 for the sparse methods. These differences can be attributed to different methods for tuning the hyperparameters as well as the sparsity assumption. Note that the active methods with predictive variance and gradient-based query functions provide further improvements in the approximation compared to sparse model checking without an active step.

When it comes to computation time, it has to be noted that one of the downsides of directly optimising the variational posterior parameters with respect to ELBO by gradient ascent means that choosing a step size for the optimiser is not always trivial. The step size affects the rate of convergence and can have significant effects on the computation times. Hence work into the computational robustness of the variational inference methods in the context of model checking as well as comparisons with other approximate inference methods like sparse expectation propagation are a direction of further research.

5 Conclusions

In this paper we applied sparse approximation and active learning to smoothed model checking. By leveraging existing sparse approximations, we improved scalability of the inference algorithms for Gaussian process classification corresponding to the smoothed model checking problem. Additionally, we showed that by concentrating the sampling to high variance or high predictive gradient areas of the parameter space, we improved the resulting approximation compared to sparse models with uniform or grid-based sampling of model parameters. When compared to the standard smoothed model checking approach with no inducing point approximation and no active step, our method significantly speeds up the inference procedure while attempting to reduce errors inherent in sparse approximations. This aligns with the pre-existing results from active learning literature which aim to construct learning algorithms that actively query for observations in order to improve accuracy while keeping the number of observations needed to a minimum.

As further work, we aim to refine our query methods and make a comparison with other existing methods in the active learning literature. Secondly, we plan to link the choice of inducing points to the active query methods more directly. In particular, we will test if the inducing points, and perhaps the underlying kernel parameters, can be effectively reconfigured through active iterations. This would further improve the approximation to the satisfaction probability surface at parts of the parameter space where satisfaction probability undergoes change. Finally we will consider alternative kernel functions. The kernel function chosen in this paper is a standard first approach in many settings but is best suited for modelling very smooth functions — not necessarily the case with satisfaction probability surfaces for parametric CTMCs.

References

1. Gaussian processes for machine learning in julia. https://github.com/JuliaGaussianProcesses, accessed: 2021-07-05

2. U-check tool. https://github.com/dmilios/U-check, accessed: 2021-07-05
3. Agha, G., Palmskog, K.: A survey of statistical model checking. ACM Trans. Model. Comput. Simul. **28**(1), 6:1–6:39 (2018)
4. Bartocci, E., Bortolussi, L., Nenzi, L., Sanguinetti, G.: System design of stochastic models using robustness of temporal properties. Theor. Comput. Sci. **587**, 3–25 (2015)
5. Bortolussi, L., Milios, D., Sanguinetti, G.: U-check: Model checking and parameter synthesis under uncertainty. In: Campos, J., Haverkort, B.R. (eds.) Quantitative Evaluation of Systems, 12th International Conference, QEST 2015, Madrid, Spain, September 1-3, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9259, pp. 89–104. Springer (2015). https://doi.org/10.1007/978-3-319-22264-6_6
6. Bortolussi, L., Milios, D., Sanguinetti, G.: Smoothed model checking for uncertain continuous-time markov chains. Inf. Comput. **247**, 235–253 (2016)
7. Bortolussi, L., Silvetti, S.: Bayesian statistical parameter synthesis for linear temporal properties of stochastic models. In: TACAS 2018,. Lecture Notes in Computer Science, vol. 10806, pp. 396–413. Springer (2018)
8. Bui, T.D., Nguyen, C.V., Turner, R.E.: Streaming sparse gaussian process approximations. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017. pp. 3299–3307 (2017)
9. Csato, L., Opper, M.: Sparse online gaussian processes. Neural Computation **14**, 641–668 (2002)
10. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: FORMATS 2010. Proceedings. Lecture Notes in Computer Science, vol. 6246, pp. 92–106. Springer (2010)
11. Hernandez-Lobato, D., Hernandez-Lobato, J.M.: Scalable gaussian process classification via expectation propagation. In: Gretton, A., Robert, C.C. (eds.) Proceedings of the 19th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 51, pp. 168–176. PMLR (2016)
12. Jha, S.K., Clarke, E.M., Langmead, C.J., Legay, A., Platzer, A., Zuliani, P.: A bayesian approach to model checking biological systems. In: Degano, P., Gorrieri, R. (eds.) Computational Methods in Systems Biology, 7th International Conference, CMSB. Lecture Notes in Computer Science, vol. 5688, pp. 218–234. Springer (2009). https://doi.org/10.1007/978-3-642-03845-7_15
13. Kwiatkowska, M.Z., Norman, G., Parker, D.: Stochastic model checking. In: SFM. Lecture Notes in Computer Science, vol. 4486, pp. 220–270. Springer (2007)
14. Legay, A., Lukina, A., Traonouez, L., Yang, J., Smolka, S.A., Grosu, R.: Statistical model checking. In: Steffen, B., Woeginger, G.J. (eds.) Computing and Software Science - State of the Art and Perspectives, Lecture Notes in Computer Science, vol. 10000, pp. 478–504. Springer (2019). https://doi.org/10.1007/978-3-319-91908-9_23
15. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: FORMATS/FTRTFT 2004, Grenoble, France, September 22-24, Proceedings. Lecture Notes in Computer Science, vol. 3253, pp. 152–166. Springer (2004)
16. Minka, T.P.: Expectation propagation for approximate bayesian inference. In: Breese, J.S., Koller, D. (eds.) UAI: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence. pp. 362–369. Morgan Kaufmann (2001)
17. Rackauckas, C., Nie, Q.: Differentialequations.jl–a performant and feature-rich ecosystem for solving differential equations in julia. Journal of Open Research Software **5**(1) (2017)
18. Rasmussen, C.E., Williams, C.K.I.: Gaussian processes for machine learning. Adaptive computation and machine learning, MIT Press (2006)

19. Santner, T.J., Williams, B.J., Notz, W.I.: The Design and Analysis of Computer Experiments. Springer series in statistics, Springer (2003)
20. Sen, K., Viswanathan, M., Agha, G.: Statistical model checking of black-box probabilistic systems. In: Alur, R., Peled, D.A. (eds.) Computer Aided Verification, 16th International Conference, CAV. Lecture Notes in Computer Science, vol. 3114, pp. 202–215. Springer (2004). https://doi.org/10.1007/978-3-540-27813-9_16
21. Settles, B.: From theories to queries. In: Active Learning and Experimental Design workshop, In conjunction with AISTATS. JMLR Proceedings, vol. 16, pp. 1–18. JMLR.org (2011), http://proceedings.mlr.press/v16/settles11a/settles11a.pdf
22. Settles, B.: Active Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers (2012). https://doi.org/10.2200/S00429ED1V01Y201207AIM018
23. Titsias, M.K.: Variational learning of inducing variables in sparse gaussian processes. In: Dyk, D.A.V., Welling, M. (eds.) Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS. JMLR Proceedings, vol. 5, pp. 567–574 (2009)