INTERGP: SYMBOLIC EXTENSION OF GAUSSIAN PROCESSES TO SET EVOLUTION

Anonymous authors
Paper under double-blind review

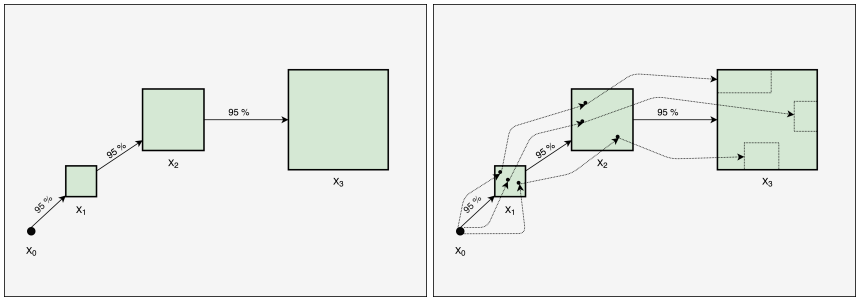
ABSTRACT

We present InterGP a symbolic extension of Gaussian processes over sets. Gaussian Process (GP) models have recently emerged as a promising technique for safe reinforcement learning. These models provide fairly accurate prediction of evolution of system dynamics with probabilistic guarantees for a given fixed state. In many real world applications, we are interested in learning dynamics model from a set of initial states and not just a fixed state. A naive application of GPs to randomly selected points or the extreme points of the initial set are not sufficient to construct dynamics model with probabilistic guarantees. In this paper, we address this problem by developing a novel extension of GPs to model the evolution of sets. We present experimental evaluation of the proposed technique on a MuJoCo environment and demonstrate the effectiveness of our approach.

1 INTRODUCTION

Reinforcement learning has emerged as an effective approach for online continual adaptation of systems in unknown environments. But their adoption in safety-critical and high-security domains is inhibited by possibly catastrophic failures during exploration in reinforcement learning. More recently, safe model-based reinforcement learning (Berkenkamp et al. (2017)) have been proposed that employ Gaussian Processes (GPs) for modeling the evolution of a dynamical system. Given a state of the system, GPs can be used to predict the distribution of next state (state after some time t). In this paper, we extend Gaussian processes to model evolution from a set of states and not just a single state. Hence, our model can be used to learn flow-pipe approximation Chutinan & Krogh (1998); Chen et al. (2012) of dynamical models. This enables less conservative prediction of safe states over multiple time-steps.

Before formally defining the problem, we introduce the notation used in rest of the paper. Let $S \subseteq \mathbb{R}^n$ be the space of possible states. Let $(X_0, \ldots, X_k, \ldots)$ be a Markov chain in S, representing a trajectory. We denote $(x_0, \ldots, x_k) \in S^{k+1}$ a potential trajectory over $k+1$ timesteps, and for $i \in \mathbb{N}$, $S_i \subseteq S$ a confidence set for X_i. Let a and b be natural numbers. We define $X_{a:b} := (X_a, \ldots, X_b)$, $x_{a:b} = (x_a, \ldots, x_b)$, $S_{a:b} = S_a \times \cdots \times S_b$. We use integer exponents with the previous notations to signify that we project along dimension i of S, such as X_k^i, x_k^i, S_k^i.



(a) A simplified representation of what we aim to achieve.

(b) InterGP symbolically covers every possible trajectory in the confidence sets.

We assume the state is fully observable. Typically, the dynamics is given a function f such that $x_{t+1} = f(x_t, u_t)$ where x_t is the state (observation) at the current timestep, x_{t+1} the state at

the next timestep, and u_t the action taken by the agent. We drop the control input u_t from being explicitly specified in the dynamics function for ease of presentation. Our focus is on set-theoretic GPs, and so, we model the dynamics simply as $x_{t+1} = f(x_t)$ without loss of generality. Given an integer k and a current state $x_0 \in S$, we want to predict the next k states, namely $(x_1, x_2, \ldots, x_k) = (f(x_0), f(f(x_0)), \ldots))$. Since it is improbable to have a perfect model of f, the error will be compounded, and a prediction can be arbitrarily bad for k sufficiently large. We want to predict a confidence set for each step (S_1 for the first step, S_2 for the second state, etc.) such that $(S_1, \ldots, S_k)$ contains the trajectory $(x_1, \ldots, x_k)$ with high probability. Formally, given a sequence of sets of states $S_1, \ldots, S_k$, we look at

$$P_k = P[X_1 \in S_1, \ldots, X_k \in S_k \mid X_0 = x_0] = P[X_{1:k} \in S_{1:k} \mid X_0 = x_0]$$

This is illustrated on figure 1a. We formally define the problem below:

Problem 1 (Computation of a lower bound on the probability of a confidence trajectory)**.** *Given a GP trained on one-step dynamics, an initial state $x_0 \in S$ and a sequence of sets of states $(S_1, \ldots, S_k)$, compute $p \in [0, 1]$ such that*

$$P[X_{1:k} \in S_{1:k} \mid X_0 = x_0] \geq p$$

Problem 2 (Synthesis of a confidence trajectory)**.** *Given a GP trained on one-step dynamics, an initial state $x_0 \in S$ and $p \in [0, 1]$, find $(S_1, \ldots, S_k)$ such that*

$$P[X_{1:k} \in S_{1:k} \mid X_0 = x_0] \geq p$$

Girard et al. (2003) and Candela et al. (2003) present an approximate solution to this problem. The authors analytically compute the distribution of a prediction from the GP when x_t is not a single state but a Gaussian random variable. The next state x_{t+1} is not Gaussian in general, but they are able to approximate its mean and variance. Then, the distribution of x_{t+1} is approximated with a normal distribution using the analytically computed mean and variance. This provides a way to propagate the uncertainty of the prediction, but this doesn't give any theoretical guarantees since it is based on successive approximations, this just gives a coarse estimate of the uncertainty. On the contrary, we have a more conservative but theoretically sound approach of uncertainty propagation, which allows us to provide real guarantees based on the GP hypothesis. This, in turn, can be used to enable safe reinforcement learning.

In this work, we make the following contributions:

- We present InterGP, an extension of GP which can formally deal with sets as input and output. InterGP is also able to deal with successive predictions to predict trajectories, and can iteratively construct a sequence of confidence sets (problem 2).
- We eventually present an example result of InterGP trajectory prediction on the inverted pendulum environment from MuJoCo (Todorov et al. (2012)).

2 Approach

Gaussian processes (GPs) (Rasmussen & Williams (2005)) are stochastic processes (collections of random variables indexed by time or space), such that every finite collection of those random variables has a multivariate normal distribution, i.e. every finite linear combination of them is normally distributed. A GP is formally the joint distribution of all those (infinitely many) random variables, but it is easier (and equivalent) to think of GPs as probability distributions over functions. GPs can be used to learn models of functions, based on a measure of the similarity between points and previous observations. Let f be the function we want to learn, and $\hat{f}$ the learned model. The measure of similarity is a covariance kernel k, such that for two points x_1 and x_2, $k(x_1, x_2) = Cov(\hat{f}(x_1), \hat{f}(x_2))$. It is used to predict the values of unseen points, and represents an estimate of the smoothness of the function. The prediction is defined by its mean and its variance: $\hat{f}(x) \sim \mathbb{N}(\mu(x), \sigma^2(x))$. There exists different classes of covariance kernels. The most popular class is the squared exponential, where the covariance between two points is the inverse of an exponential function of the distance between the points. One way to train GPs is to find the best parameters for the given covariance kernels class given the available data, for instance by maximizing the log-likelihood.

We make the following assumptions in InterGP.

Assumption 1 (Deterministic dynamics)**.** *We assume the existence of a fully deterministic function f such that the dynamics of the system are determined by $x_{t+1} = f(x_t)$.*

Proposition 1 (Conditional independence)**.** *If the dynamics are deterministic, then the components of $X_{t+1} = [X_{t+1}^1, \dots, X_{t+1}^n]^T$ are mutually conditionally independent on $X_t = x_t$.*

Indeed, knowing exactly X_t and some components of X_{t+1} is equivalent to just knowing exactly X_t. It allows us, at each step of the process, to work independently in every dimension, and combine the results in the end. We thus reason about $x_{t+1}^i = f^i(x_t)$ in the following subsections.

Definition 1 (Confidence interval)**.** *$[a, b]$ is a confidence interval for a random variable X with probability p if $P[a \leq X \leq b] \geq p$.*

InterGP tackles the described problems by using Gaussian Processes to model the one-step dynamics. InterGP is an iterative process, which looks at the steps one by one. It is easy to create a confidence set $S_1 \subseteq S$ for X_1 knowing that $X_0 = x_0$, and using proposition 1, by creating for each dimension separately a confidence interval, and then combining these confidence intervals into hyperrectangles. However, it is more challenging to create a confidence set S_2 for X_2 knowing that $X_0 = x_0$ and $X_1 \in S_1$, since the input is not a single point anymore but a set. Furthermore, the Gaussian Processes have to take into account the previous predictions, such as $X_1 \in S_1$, when trying to predict X_2. InterGP tackles this issue by symbolically covering every trajectory in the successive predicted confidence sets, as illustrated on figure 1b and as described below. Eventually, this approach of having one confidence set for every trajectory boils down to a problem of finding the union of these confidence sets. We describe below how to choose the confidence intervals for each trajectory in each dimension in order to minimize the resulting confidence set after the union, instead of just taking the union of confidence intervals centered around the means.

Notations 1 (CDF and PPF)**.** *The CDF and PPF function are defined such that $CDF^i_{x_{0:k-1}}(b) = P\left[X_k^i \leq b \mid X_{0:k-1} = x_{0:k-1}\right]$ and $P\left[X_k^i \leq PPF^i_{x_{0:k-1}}(p)\right] = p$.*

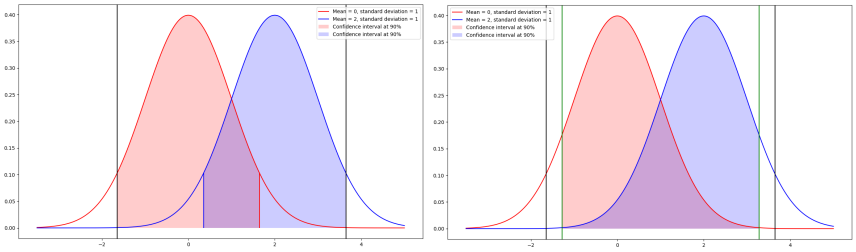
Definition 2 (Risk allocation)**.** *A risk allocation is $n \times k$ values $(p_j^i)_{i \in [1..n], j \in [1..k]}$, such that*

$$\forall i \in [1..n], \forall j \in [1..k], \forall x_{0:j-1} \in S_{0:j-1}, P[X_j^i \in S_j^i \mid X_{0:j-1} = x_{0:j-1}] \geq p_j^i$$

Algorithm 1: Outline of InterGP algorithm

Data: x_0, k, risk allocation $(\tilde{p}_j^i)_{(j,i) \in [1..k] \times [1..n]}$, GP trained on one-step dynamics
Result: $(S_1, \dots, S_k)$
$S_0 = \{x_0\}$;
for $j = 1$ *to* k **do**
  for $i = 1$ *to* n **do**
    $\tilde{a} = \inf_{x_{0:j-1} \in S_{0:j-1}} PPF^i_{x_{0:j-1}}(1 - \tilde{p}_j^i)$;
    $\tilde{b} = \sup_{x_{0:j-1} \in S_{0:j-1}} PPF^i_{x_{0:j-1}}(\tilde{p}_j^i)$;
    $[a, b](\_) = maxintersection(\_, \tilde{a}, \tilde{b}, \tilde{p}_j^i)$;
    $S_j^i = [\inf_{x_{0:j-1} \in S_{0:j-1}} a(x_{0:-1j}); \sup_{x_{0:j-1} \in S_{0:j-1}} b(x_{0:j-1})]$;
  $S_j = S_j^1 \times \dots \times S_j^n$;
Return $(S_1, \dots, S_k)$

As shown with notations 1, InterGP uses the distribution of X_j^i knowing that $X_{0:j-1} = x_{0:j-1}$. The next subsection briefly describes how to compute this distribution. InterGP first loops on the time steps (1 to k) to iteratively compute confidence sets, and inside the outer loop it loops on each dimension of the step. $\tilde{a}$, $\tilde{b}$ and the *maxintersection* algorithm aim at choosing a confidence set for each possible trajectory to minimize the resulting union of the confidence sets, and are described in a following subsection. a and b are functions. Eventually, InterGP uses a risk allocation because of the modularity of its computation.



(a) Naive approach: We center both interval confidences before taking the union.

(b) Max intersection approach: We try to maximize the intersection.

Figure 2: Comparison of two approaches to find a 90% interval confidence for two standard normal laws.

Theorem 1 (Soundness of InterGP). *If $(S_1, \dots, S_k)$ is a solution given by InterGP with the risk allocation $(p_j^i)_{i\in[1..n],j\in[1..k]}$, then*

$$P[X_{1:k} \in S_{1:k} \mid X_0 = x_0] \geq \prod_{j=1}^{k} \prod_{i=1}^{n} p_j^i$$

InterGP and the *maxintersection* algorithm need to compute $PPF^i_{x_{0:j-1}}$ and $CDF^i_{x_{0:j-1}}$, which require knowing the distribution of X_j^i given a trajectory. Given a trajectory $X_{0:j-1} = x_{0:j-1}$, X_j^i follows a normal law. Indeed, let $Y^i := [X_1^i, \dots, X_j^i]^T = [f^i(X_0), \dots, f^i(X_{j-1})]^T$, Y^i knowing that $X_{0:j-1} = x_{0:j-1}$ follows a normal whose mean and variance can be easily computed with the Gaussian Processes. Then, the distribution of X_j^i is derived from the distribution of Y^i conditioned on $X^i_{1:j-1} = x^i_{1:j-1}$, which is also Gaussian.

This approach is illustrated in finding a confidence interval for two Gaussian distributions that differ in mean on Figure 2a. Indeed, we are not restricted to intervals centered around the mean, we can easily translate the intervals to the right or the left. We proceed in two steps. We first compute two theoretical limits on the bounds of such a confidence interval, we then use them to maximize the intersection between normal laws. This approach is illustrated in figure 2b.

Let $\tilde{a} = \inf_{x_{0:j-1}\in S_{0:j-1}} PPF^i_{x_{0:j-1}}(1 - \tilde{p}_j^i)$ and $\tilde{b} = \sup_{x_{0:j-1}\in S_{0:j-1}} PPF^i_{x_{0:j-1}}(\tilde{p}_j^i)$. $\tilde{a}$ and $\tilde{b}$ are respectively an upper bound and a lower bound for the bounds of any confidence interval, that is to say any confidence interval will be included in $[\tilde{a}, \tilde{b}]$. For every trajectory x, *maxintersection* tries first to fill $[\tilde{a}, \tilde{b}]$ to get a confidence interval. If this interval is not sufficient, *maxintersection* grows symmetrically this interval around the mean of the distribution of the dimension i of the next state, until the interval is sufficient for the probability p. To work, *maxintersection* requires $p \geq 0.5$. The algorithm is formally described below.

Notations 2. *For a trajectory x, we denote $\mu^i(x)$ and $\sigma^i(x)$ the mean and the standard deviation of the normal law followed by the dimension i of the next state.*

Proposition 2. *The solution given by maxintersection is at least as good as the union of the confidence intervals centered around the means of each distribution for each trajectory in the input. Furthermore, if at least two inputs give different means in the distribution of the next state, then maxintersection computes a smaller resulting confidence interval.*

Problem 1 is straightforward to tackle since we know how to compute the distributions of the states. The algorithm is described below.

Remark 1. *Algorithm 3 is more precise than the synthesizing algorithm 1, meaning that applying algorithm 3 on a solution of algorithm 1 gives a result p such that $p \geq \prod_{k'=1}^{k} \prod_{i=1}^{n} p_{k'}^i$, and the inequality is generally strict.*

Algorithm 2: Max intersection algorithm

Data: A trajectory x, a dimension i, $\tilde{a}$ and $\tilde{b}$ previously computed, a probability p
Result: Confidence interval with probability p
$\tilde{p} := CDF_x^i(\tilde{b}) - CDF_x^i(\tilde{a})$;
if $\tilde{p} \geq p$ **then** return $\left[\tilde{a}, \tilde{b}\right]$;
$\Delta := \max(|\mu^i(x) - \tilde{a}|, |\tilde{b} - \mu^i(x)|)$;
if $CDF_x^i(\mu^i(x) + \Delta) - CDF_x^i(\mu^i(x) - \Delta) \geq p$ **then**
  if $|\mu^i(x) - \tilde{a}| < |\tilde{b} - \mu^i(x)|$ **then** return $\left[PPF_x^i(CDF_x^i(\tilde{b}) - p), \tilde{b}\right]$;
  else return $\left[\tilde{a}, PPF_x^i(CDF_x^i(\tilde{a}) + p)\right]$;
else
  Return $\left[PPF_x^i(\frac{1-p}{2}), PPF_x^i(\frac{1+p}{2})\right]$

Algorithm 3: Algorithm for the dual problem

Data: x_0, k, $S_k^i = [a_k^i, b_k^i]$, trained GP
Result: Lower bound on $P[X_{1:k} \in S_{1:k} \mid X_0 = x_0]$
$p = 1$;
for $j = 1$ *to* k **do**
  for $i = 1$ *to* n **do**
    $p_k^i = \inf_{x_{1:k-1} \in S_{1:k-1}} \left[CDF_{x_{1:k-1}}(b_k^i) - CDF_{x_{1:k-1}=}(a_k^i)\right]$;
    $p = p * p_k^i$;
Return p

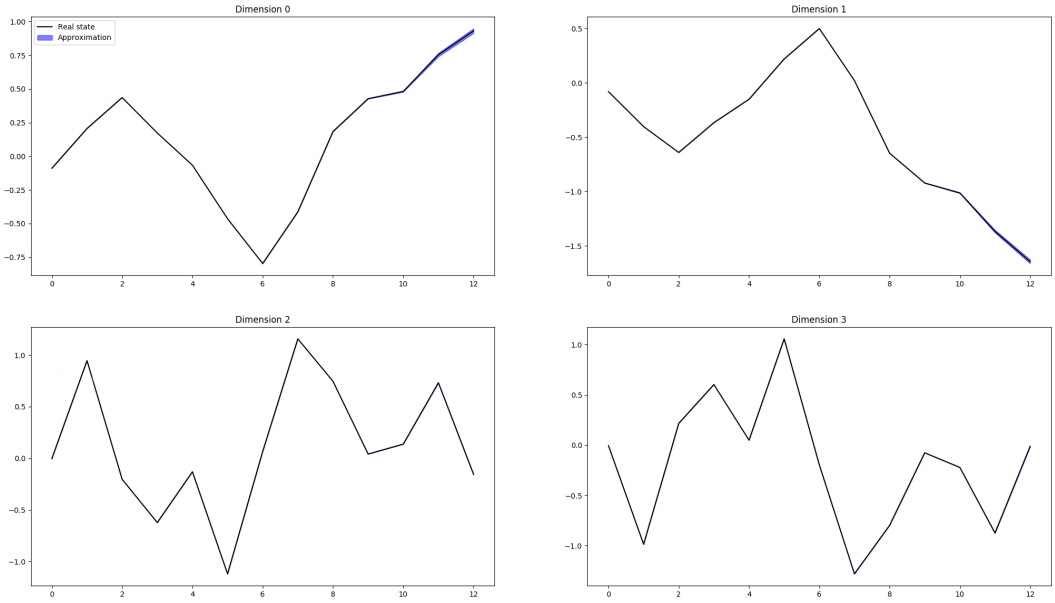


Figure 3: Reachability on Inverted Pendulum (MuJoCo), $p = 99\%$, 500 training samples.

3 EXPERIMENTS AND RESULTS

InterGP was implemented using Numpy (Oliphant (2006)) and Scikit-Learn (Pedregosa et al. (2011)) for the GPs, Scipy for the minimizers (Jones et al. (2001–)), and Matplotlib (Hunter (2007)) to visualize the results. An example of a multiple-step ahead is shown on figure 3, on the inverted pendulum environment from MuJoCo (which is one environment from MuJoCo where InterGP performs the best since it has only four dimensions). The agent in this environment is a cart whose aim is to keep a pole balanced, while laterally staying close to the center. The agent can choose to go either left or right, and observes a state with four dimensions: the algebraic lateral distance d from the center of the cart to the center of the environment, the vertical oriented angle θ of the pole, and the time derivatives of both, $\dot{d}$ and $\dot{\theta}$. We collected 500 training samples (x_t, u_t, x_{t+1}) from a random policy to train the GPs on one-step dynamics. We then ran again the random policy which ended after 12 timesteps, and used InterGP on this trajectory, with a risk allocation of 0.9975 for each step and each dimension, which gives a lower bound above 0.88 on the probability for the trajectory to be in the generated confidence sets.

The four dimensions are represented on the figure, where the abscissa is the timestep and the ordinate is the normalized value of the state. The black curves represent the true state, and the areas filled with blue represent the confidence sets. As can be seen, InterGP is really accurate up to 10 timesteps, where it is almost impossible to distinguish between the true state and the confidence set. After 10 timesteps, we begin to see the confidence sets grow, and propagate to the other dimensions as well. InterGP should be further investigated on MuJoCo and OpenAI Gym (Brockman et al. (2016)) environments, and rigourously benchmarked against other forecasting methods which don't provide formal guarantees.

4 DISCUSSION ON ASSUMPTIONS AND LIMITATIONS

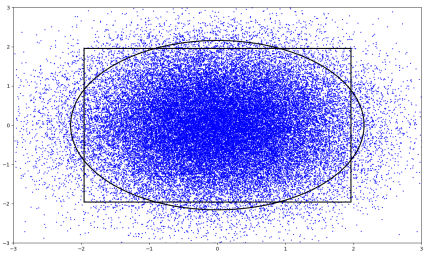
4.1 ITERATIVE COMPUTATION

One advantage of our iterative approach is that we not only compute a confidence set at k time steps in the future, but we also have a confidence set for every step. This allows more flexibility, but is a limitation in the sense that our prediction at k time steps could be improved. Indeed, when we use a confidence set S_t to compute a confidence set S_{t+1}, we have "forgotten" the distribution that led to S_t (it is often the case that the closer to the mean a trajectory is, the more "probable" this trajectory is). The methods described in this work could in theory be modified and applied to compute directly a confidence set at k time steps, but this seems computationally difficult. Another idea could be to use the iterative approach described here, and try to optimize the risk allocation. The only important aspect when looking at the horizon is the value of the product of the probabilities. For example, one could achieve a better probability at the horizon if one put lower probabilities at the beginning, to be more accurate, since the error is compounded over time.

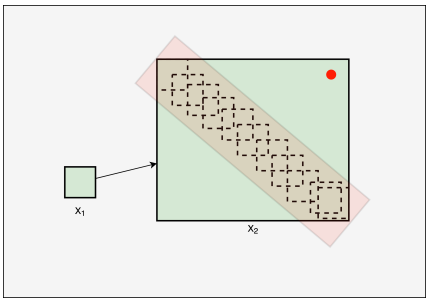
4.2 HYPERRECTANGLES (AND ELLIPSOIDS)

"Shape" in this context refers to the geometric shapes of set we use to generate our confidence sets (hyperrectangles for example). InterGP can work with different shapes. It can take any shape as input, even though convex shapes are better for optimization. For every distribution of the next state given a trajectory as input, InterGP generates a confidence set and then combines these sets into a confidence set that works for all inputs. For both the single confidence sets and the resulting one, we use hyperrectangles, since they are easy to generate (with independent confidence intervals) and easy to combine: The resulting hyperrectangle can be expressed with only $\min$ and $\max$ functions.

However, it seems that InterGP could also use non-rotated ellipsoids. Indeed, as shown on figure 4a, when the components of the next state are following mutually independent normal laws, the confidence set with minimum volume is a non-rotated ellipsoid. Using a χ^2-distribution with n degrees of freedom, it is easy to compute a good confidence ellipsoid for a given point x, but the hard part is to combine them. There only exist approximation algorithms to find the minimum volume ellipsoid covering a union of ellipsoids (Sun & Freund (2004)), and these algorithms are too complicated to be symbolically computed. One easy approximation could be to find the smallest hyperrectangle con-



(a) Comparison of two 90% confidence shapes, a rectangle and an ellipsoid.



(b) One limit of our approach. In this example, the red point is a state that doesn't make physical sense. The prediction of X_3 will explode. The tilted transparent red rectangle could be a solution to tackle this issue.

taining the ellipsoids, and then to find an ellipsoid covering this hyperrectangle, but this obviously doesn't bring any advantage on directly using hyperrectangles.

One limit of using a non-rotated shape can be the following. Suppose the dimensions of a state in an environment are highly dependent, for example as represented on figure 4b. In this example, every input would yield a confidence hyperrectangle around the possible values. InterGP would then combine these hyperrectangles in a hyperrectangle that contains a lot of physically meaningless states, such as the red point. At the next step (to compute S_3), the GP would be clueless around this point, since there wouldn't be any training data near it, so this would yield a really high variance: S_3 would then "explode". In this particular example, rotated hyperrectangles could work. Another option could be to preprocess the available data and to project the states in a better suited space, or to use dimensionality reduction techniques.

5 Conclusion

In this work, we presented InterGP, an extension of GPs to formally deal with sets, with an application to formal multiple-step ahead forecasting. We presented some theoretical and experimental results, and discussed some assumptions and limitations. We are working on extensions of this approach. First, as discussed, one could find better confidence shapes, or find a better way to combine these shapes. Moreover, it could be interesting to learn more models, such as one-step and two-steps dynamics, and to try to combine them to achieve better results on multiple-step ahead forecasting. Another vector of improvement is the choice of the risk allocation: Is it better to put higher probabilities at the beginning or at the end of the predictions, given a fixed lower bound probability on the trajectory? Furthermore, InterGP trajectory predictions should be more closely investigated and benchmarked against other techniques, even though these other techniques don't give formal guarantees, to analyze the accuracy gap.

References

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in Neural Information Processing Systems 30*, pp. 908–918. 2017.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

J. Q. Candela, A. Girard, J. Larsen, and C. E. Rasmussen. Propagation of uncertainty in bayesian kernel models - application to multiple-step ahead forecasting. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pp. II–701, April 2003. doi: 10.1109/ICASSP.2003.1202463.

Xin Chen, Erika Abraham, and Sriram Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *2012 IEEE 33rd Real-Time Systems Symposium*, pp. 183–192. IEEE, 2012.

Alongkrit Chutinan and Bruce H Krogh. Computing polyhedral approximations to flow pipes for dynamic systems. In *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No. 98CH36171)*, volume 2, pp. 2089–2094. IEEE, 1998.

Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Agathe Girard, Carl Edward Rasmussen, Joaquin Quiñonero Candela, and Roderick Murray-Smith. Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. In *Advances in Neural Information Processing Systems 15*, pp. 545–552. MIT Press, 2003.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science Engineering*, 9(3): 90–95, May 2007. ISSN 1521-9615. doi: 10.1109/MCSE.2007.55.

Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL `http://www.scipy.org/`. [Online; accessed ¡today¿].

Travis Oliphant. *Guide to NumPy*. 01 2006.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

Peng Sun and Robert M. Freund. Computation of minimum-volume covering ellipsoids. *Operations Research*, 52(5):690–706, 2004. doi: 10.1287/opre.1040.0115. URL `https://doi.org/10.1287/opre.1040.0115`.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, Oct 2012. doi: 10.1109/IROS.2012.6386109.