Probabilistic Model Learning and Long-term Prediction for Contact-rich Manipulation Tasks

Shahbaz A. Khader\textsuperscript{1,2}, Hang Yin\textsuperscript{1}, Pietro Falco\textsuperscript{2} and Danica Kragic\textsuperscript{1}

Abstract—Learning dynamics models is an essential component of model-based reinforcement learning. The learned model can be used for multi-step ahead predictions of the state variable, a process referred to as long-term prediction. Due to the recursive nature of the predictions, the accuracy has to be good enough to prevent significant error buildup. Accurate model learning in contact-rich manipulation is challenging due to the presence of varying dynamics regimes and discontinuities. Another challenge is the discontinuity in state evolution caused by impacting conditions. Building on the approach of representing contact dynamics by a system of switching models, we present a solution that also supports discontinuous state evolution. We evaluate our method on a contact-rich motion task, involving a 7-DOF industrial robot, using a trajectory-centric policy and show that it can effectively propagate state distributions through discontinuities.

I. INTRODUCTION

Model learning, or learning dynamics models from data, is a key element of model-based reinforcement learning (MBRL). MBRL is attractive due to its better sample efficiency over model-free methods [1]. The learned forward dynamics model can be used as a simulator for evaluating the performance of a candidate policy. In this process, also called as long-term prediction, successive predictions from the model and the policy are cascaded in order to predict a trajectory distribution for a time horizon of interest. Such recursive predictions are susceptible to error accumulation and can deviate into regions where the model has seen less training data. When the uncertainty due to limited training data is also incorporated into the probabilistic model, further improvement in sample efficiency can be achieved [2]. Therefore, accurate model learning and long-term prediction of such uncertainty aware models is important for MBRL.

We consider the case of contact-rich manipulation tasks, such as assembly, in static and rigid environments. Although model learning can be achieved using any standard regression methods such as artificial neural networks (ANN) or Gaussian process (GP), the complex nature of contact dynamics can lead to poor performance. The dynamics for contact-rich manipulation is characterized by smooth regions representing different contact situations separated by discontinuities (Fig. 1b). In such situations, the mixture of experts (ME) [3] strategy would be more appropriate, where a number of experts (or modes in the case of dynamics) that model the different regions switch between one another. However, such an approach introduces a problem for long-term prediction. A straightforward long-term prediction using a standard ME model will not be able to handle the discontinuity in the state (velocity) evolution that can be caused by an impacting contact (Fig. 1d). Our main goal is a method that can learn a discontinuous dynamics model represented as a system of switching modes, encode uncertainty due to limited training data, and support discontinuous long-term prediction. Our problem is also related to system identification of hybrid dynamical system [4], learning switching linear dynamical system (SLDS) [5], and learning hidden Markov model (HMM) based switching models [6], but none of the existing frameworks is designed to satisfy our specific goal.

Our method relies on particle based uncertainty propagation and is intended to be used as a gradient-free option. We obtain a system of uncertainty-aware models by clustering the trial data and learning independent dynamics modes (GP). Next, we introduce a novel switching scheme that involves learned one-step-ahead mode predictor and initialization models that predicts initial state distributions of modes for achieving long-term prediction. The novel switching scheme and its associated models is what makes the important contribution of allowing discontinuous (Gaussian) state evolution. It also supports simultaneous propagation through multiple modes in the form of a mixture of Gaussians. Our experiment on a 7-DOF industrial robot shows that our method can scale to real scenarios and outperform highly flexible but general purpose model learning baselines.
II. RELATED WORK

Although model learning for robot control has been actively researched [7], learning forward dynamics models for contact-rich manipulation tasks that feature discontinuities has been less explored. Most methods address learning only inverse dynamics models, which are often used for one-step prediction. For MBRL in a contact-rich manipulation context, the learned forward model has to be evaluated for long-term prediction through discontinuities. Such a direct evaluation was lacking in all the reviewed works.

A prominent example of probabilistic model learning for MBRL is PILCO [2], in which GPs were used for model learning. Unfortunately, with squared exponential (SE) kernels, PILCO does not support learning and prediction through discontinuous dynamics. A strategy for addressing discontinuities within the GP framework is manifold GP [8]. The SE kernel function operates in a feature space, that is transformed from the input space using an ANN. However, the method was evaluated only for single step prediction. A more recent work (PETS) [9] that learns an uncertainty aware ANN model (ensemble of bootstraps), succeeded in regaining many of the desirable features of GP while retaining the expressive power and scalability of ANN. Apart from benefiting from the data efficiency of GP, our method has two main differences over the model learning approach in PETS: 1) it explicitly handles discontinuity and therefore can potentially model contacts better and 2) it has a parametric form (GMM) for multimodal state distribution instead of a particle based representation.

The mixture of experts (ME) approach [3], in which a number of local models or experts are learned together with a gating network, provides a framework for highly complex models. Inverse dynamics models were learned for contact tasks in [10] and [11] and for free motion task in [12]. McKinnon et al. in [13] used GP based Dirichlet Process Mixture Model to learn the forward dynamics model of a mobile robot. Such ME inspired approaches, and the more general GP mixture model in [14], where the current dynamics mode is inferred based on the present or past data, may be sufficient for one step prediction, but is not designed for discontinuous state propagation. To remedy this, our method introduces a one-step-ahead mode predictor, a GP based initial state predictor and a probabilistic switching scheme to support not only discontinuous dynamics but also discontinuous state evolution.

Also relevant to our work is forward model learning with multistep prediction for contact-rich tasks such as [15] and [16], in which deterministic dynamics models were learned using ANN. An uncertainty-aware approach, such as ours, can potentially help achieve greater data efficiency for MBRL [2]. Finally, Levine et al. [17] used Gaussian mixture models (GMM) to learn locally linear model priors. While each component of the GMM represented a locally linear dynamics mode, our method benefits from a more flexible nonlinear GP model.

III. PROBLEM FORMULATION

In contact-rich manipulation, the robot-environment interaction is characterized by contact situations such as sliding, sticking, slipping, or motion constrained by obstacles. Our modeling assumptions are: 1) smooth dynamics within a contact situation, 2) impacting contact causes instantaneous state change (velocities), and 3) a static and rigid environment.

The goal of model learning is, given a training dataset consisting of tuples \((x_t, u_t)\), to fit a model of the form,

\[
p(x_{t+1} | x_t, u_t) = \mathcal{N}(\mu(x_t, u_t), \Sigma(x_t, u_t)),
\]

where \(x_t = [q^T_t \ \dot{q}^T_t]^T\) is the state vector comprising of the joint positions \(q_t \in \mathbb{R}^D\) and velocities \(\dot{q}_t \in \mathbb{R}^D\), \(u_t \in \mathbb{R}^D\) is the control action taken by a policy, \(D\) is the number of axes of the manipulator, \(\mu(\cdot)\) and \(\Sigma(\cdot)\) are the mean and variance functions of the dynamics model. In contact-rich manipulation, \(\mu(\cdot)\) (and \(\Sigma(\cdot)\)) can change instantaneously with contact situations (Fig. 1b). While GP with SE kernel certainly cannot express such functions, ANN under low data regimes can also fail. The ME model is a natural framework for such cases, where a set of experts and a gate are learned to solve the regression problem. Equation (2) summarizes the standard ME model for forward dynamics, where \(i\) represents an expert (or mode) and \(\theta\)'s the model parameters.

\[
p(x_{t+1} | x_t, u_t) = \sum_{i=1}^K p(i | x_t, u_t, \theta_g) p(x_{t+1} | i, x_t, u_t, \theta_e)
\]

The gate (or mode predictor) \(p(i | \cdot)\) splits the input space of the regression model into a number of regions. For each region, an expert \(p(x_{t+1} | \cdot)\) is trained. If the most likely expert is selected (hard switching) instead of averaging, discontinuities can be represented more accurately.

In long-term prediction, we seek the prediction of the state distributions \(p(x_1 | \pi), p(x_2 | \pi), ..., p(x_T | \pi)\), starting from an initial state distribution \(p(x_0)\), where \(\pi = p(u_t | x_t)\) is a policy that is under evaluation and \(T\) is the time horizon of interest. Obtaining long-term prediction by cascading the model in (2) will only work if during switching one expert (plus the policy) succeeds in driving the state variable into the next experts input region. In other words, there has to be a continuous evolution of state regardless of dynamics switching (Fig. 1c). Unfortunately, this is not the case for contact-rich manipulation, in which the state variables can undergo instantaneous changes in velocities during contact (Fig. 1d).

In MBRL, long-term prediction that encodes the subjective uncertainty due to limited training data (epistemic uncertainty), in addition to the inherent stochasticity in the system (aleatoric uncertainty), can be advantageous [2]. Furthermore, contacts may give rise to discrete events, such as stick or slip, that may be probable for the same policy. Therefore, another desirable feature is the ability to propagate multiple state paths.

We define the problem as: 1) representation of contact-rich dynamics by a system of switching models, 2) long-term prediction with discontinuous state evolution, 3) encoding
epistemic and aleatoric uncertainties, and 4) propagation of a multimodal distribution that reflects multiple probable dynamics modes.

**IV. MODEL LEARNING AND LONG-TERM PREDICTION**

Our method has two main aspects: model learning (Alg. [1]) and long-term prediction (Alg. [2]). They are also visualized in Fig. 2. The ME inspired system of switching dynamics models is realized by learning a set of GP dynamics models. GP regression is the most established method for encoding both epistemic and aleatoric uncertainties. Our method exploits discontinuities in state data of contact-rich manipulation by first clustering the dataset (in state space or its equivalent) and then fitting dynamics modes using individual clusters. The number of such models are automatically inferred from data. As a key step, we also learn a one-step-ahead mode predictor and initialization models that predict initial state distributions for switching pairs.

Long-term prediction is addressed by a novel switching scheme. The one-step-ahead mode predictor is used to determine switching probabilities. In the event of a switch, the initialization model predicts the initial state distribution of the next mode. This allows discontinuous state evolution. Probabilistic switching is introduced to manage the splitting of the state distributions caused by discontinuities. The result is an overlapping of active modes in time with states in each mode represented as a weighted Gaussian. The same mechanism also supports a continued propagation of multiple modes as mixture of Gaussians.

**A. Learning Forward Models of Switching Dynamics**

Equation (1) is the standard form for probabilistic dynamics model. We adopt an alternative form [2] that predicts the difference between the next and current states. Individual parts of our model learning can be formalized as:

\[ i_{t+1} = f_{mode, pred}(x_t, u_t) \]

\[ p(\Delta x_i | i_t, x_t, u_t), \quad x_{t+1} = x_t + \Delta x_i \]

\[ p(x_{t+1} | i_t, i_{t+1}, x_t, u_t), \quad i_t \neq i_{t+1} \]

The variable \( i_t \) represents a mode at time \( t \). Equation (3) represents a deterministic one-step-ahead mode predictor. Equation (4) represents the dynamics model for mode \( i_t \). Equation (5) represents the initialization model that predicts the initial state distribution of a new mode \( i_{t+1} \) when transitioned from the current mode \( i_t \).

The model learning procedure is summarized in Alg. [1] and also depicted in the upper half of Fig. 2. The dataset obtained from a batch of trials is first clustered in the state space (or its equivalent) using the Dirichlet Process Gaussian Mixture Model [18] (DPGMM) method, which automatically infers the number of clusters (modes), \( K \). Fig. 2 symbolically depicts the clustering in a two-dimensional state space. We maintain the sequential order of the dataset and the cluster labels \( z \), which will be exploited elsewhere in the algorithm. Next, we train a multioutput GP dynamics model for each cluster with its respective data \( \{D_{x,i}^{u,t}, D_{x,i}^{\Delta x} \} \). Each output dimension is modeled as an independent GP. Although the learned GP model is (4), for representational simplicity we refer to the full model in (1) as \( G_{i}^{D_{x,i}^{\Delta x}} \). The initialization models in (5) are also learned with GP regression. For every mode switching \( i \rightarrow j \) that is found in the dataset, a multioutput GP (independent outputs), \( G_{i}^{D_{x,i}^{\Delta x}} \), is learned for predicting the initial state distribution of mode \( j \) conditioned on the state action pair \( (x_t, u_t) \) in mode \( i \) at the instant of transition. The training data, \( \{D_{x,i}^{u,t}, D_{x,i}^{\Delta x} \} \), for each initialization model is extracted from the clustered dataset.
Algorithm 1 Model learning for switching dynamical system

Require: Temporarily ordered data set \( \mathcal{D} \)

1: \( z, K \leftarrow \text{DPGMM cluster on } \mathbb{D}^n \) \( \triangleright \) cluster on state data
2: for each cluster \( i = 1, ..., K \) do
3: \( \mathcal{GP}_{i}^{\text{mode}} \leftarrow \text{GPR}(\mathbb{D}^{n|x}, \mathbb{D}^{n|x}) \) \( \triangleright \) Mode \( i \)
4: for each mode switch \( i \rightarrow j \) do
5: \( \mathcal{GP}_{ji}^{\text{init}} \leftarrow \text{GPR}(\mathbb{D}^{n|x}, \mathbb{D}^{n|x}) \) \( \triangleright \) Init model \( ji \)
6: \( \mathcal{SV}_{\mathcal{J}^{\text{pred}}} \leftarrow \text{SVM}_{\text{train}}(\mathbb{D}^{n|x}, z') \) \( \triangleright \) Mode predictor

that had the sequential order of data preserved. In the final step, the one-step-ahead mode predictor model \( \mathcal{SV}_{\mathcal{J}^{\text{pred}}}(. , z') \), which approximates \( \mathcal{P} \), is obtained by training a multiclass support vector machine (SVM) classifier using the dataset \( (\mathbb{D}^{n|x}, z') \), where \( \mathbb{D}^{n|x} \) and \( z' \) consists of arrays of \( (x_t, u_t) \) and \( z_{t+1} \), respectively. The SVM hyperplane (in \( x \)-space) is symbolically depicted in the upper right part of Fig. 2. This is another use of the sequential order of the dataset.

The motivation to use DPGMM [18] is to be independent of the number of modes. In practice, we use the Bayesian Gaussian mixture, a variational inference based implementation available at [19], that requires an upper bound on the number of clusters, \( K_{\text{max}} \). Clustering dynamics data to model piecewise smooth dynamics has been previously tried in [17] where a regular GMM clustering in the joint input-output space of the model was used. For our robot experiment, we got best results for clustering in the Cartesian state space, with the state represented by positions and velocities of three non-collinear points defined at the end-effector. To prevent spurious switching events, we implemented a heuristic that reassigns all data points that form small isolated groups (\( <= 2 \)) along a trajectory.

GP regression is a Bayesian non-parametric method that is well suited for our needs. A GP is a distribution of functions \( f \sim \mathcal{GP}(m, k) \) completely defined by a mean function \( m \) and covariance function \( k \). In our case, \( m \equiv 0 \) and \( k \) is the standard SE function with automatic relevance determination (ARD). The GP model provides uncertainty estimates for predictions and in our case allows independent noise level estimation for each output dimension of each mode. The GP models are trained by optimizing the log marginal likelihood.

SVM has been previously used for gating in a standard ME model in [10]. In our model, \( \mathcal{SV}_{\mathcal{J}^{\text{pred}}} \), has the crucial difference of being a one-step-ahead predictor.

B. Uncertainty Propagation in Forward Prediction

Long-term prediction is achieved by repeating one-step predictions recursively. But, it involves propagating uncertain inputs through the learned models in \( \mathcal{P} \) and also the policy \( p(u_i|x) \). We focus on this process which is also shown in the bottom half of Fig. 2.

We derive a probabilistic mode predictor (MC-SVM) from the previously learned deterministic model, \( \mathcal{SV}_{\mathcal{J}^{\text{pred}}}(. , z') \), by using a simple Monte Carlo approach. A number of samples are drawn from the input distribution \( p(x_t, u_t) \) and the relative frequencies of the predicted classes are approximated as a discrete probability distribution \( p(i_{t+1}) \). MC-SVM is depicted in the lower left section of Fig. 2 with dots used to represent the particles. The MC process is represented with the operator \( \mathcal{MC}(f(x), p(x)) \), where \( f(x) \) and \( p(x) \) are the predictor and the input distribution, respectively.

We chose another particle based method [20] that is based on the unscented transform (UT) method [21] for the GP models \( \mathcal{GP}_{i}^{\text{mode}} \) and \( \mathcal{GP}_{ji}^{\text{init}} \). An analytic but less computationally efficient alternative is moment matching [22]. We also exploit the particle based approach to split state-action distributions during switching. Following the outcomes of the mode predictor, for each pair \( (i_t, i_{t+1}) \), a corresponding state-action distribution \( p(x_t, u_t|i_t, i_{t+1}) \) is approximated using the particles involved in the transition \( (i \rightarrow i_{t+1}) \). During a mode switch, when both the original mode and the next mode are simultaneously active for a few time steps, \( p(x_t, u_t|i_t, i_{t+1}) \) is propagated through \( \mathcal{MC} \) for \( i_t = i_{t+1} \) and \( \mathcal{GPR} \) for \( i_t \neq i_{t+1} \). Through this splitting the prediction and training data are made consistent.

UT is a particle based method that is used to propagate an input distribution through a nonlinear function. It is also used on the policy. As introduced in [20], the GP prediction variance for the mean particle is added to the transformed input distribution. A similar approach is taken for the policy \( \pi \) and its intrinsic variance. The cross-covariance between \( x_t \) and \( u_t \) is obtained in a manner similar to the variance and used to form \( p(x_t, u_t) \). Fig. 2 shows the cases for the GP based models with the UT particles indicated as crosses. Similar to the case with MC, we represent the UT process with the operator \( UT(f(x), p(x)) \).

The forward prediction is summarized in \( \mathcal{P} \) where \( \mathcal{P}_d \) and \( \mathcal{P}_e \) are valid for \( i_t = i_{t+1} \) and \( i_t \neq i_{t+1} \), respectively.

\[
\begin{align*}
\mathcal{P}(u_t|x_t, i_t) &\leftarrow \mathcal{UT}(\pi, p(x_t|i_t)) \\
\mathcal{P}(x_{t+1}|u_t, i_t) &\leftarrow \mathcal{MC}(\mathcal{SV}_{\mathcal{J}^{\text{pred}}}, p(x_t, u_t|i_t)) \tag{6a} \\
\mathcal{P}(x_{t+1}|i_t, i_{t+1}, .) &\leftarrow \mathcal{MC}(\mathcal{SV}_{\mathcal{J}^{\text{pred}}}, p(x_t, u_t|i_{t+1})) \tag{6b} \\
\mathcal{P}(x_{t+1}|i_t, i_{t+1}, .) &\leftarrow \mathcal{MC}(\mathcal{SV}_{\mathcal{J}^{\text{pred}}}, p(x_t, u_t|i_t)) \tag{6c} \\
\end{align*}
\]

C. Long-term Prediction with Switching Dynamics Models

Long-term prediction is achieved by cascading the one-step prediction steps in \( \mathcal{P} \). This induces a distribution of all trajectories possible with the learned model and the current policy. For switching dynamics models that feature discontinuous state evolution, the trajectory distribution will be piecewise continuous. We refer to each piece, that evolves within a mode, as a segment (denoted by \( s \)). Because modes can be revisited, many segments can be associated with a mode but not vice versa. In our novel switching scheme, segments can overlap in time during switching, which we call as probabilistic switching, or when multiple modes are active simultaneously. An example of the latter case is stick-slip phenomenon where both cases are probable.

In Alg. 2 we drop the conditioning on \( i \) in \( \mathcal{P} \) and indicate the same by the superscript \( s \) since \( s \) completely defines \( i \). The first segment is initialized with \( p(x_0) \) and weight one. The mode predictor predicts the probabilities
Algorithm 2 Long-term prediction with switching dynamics

Require: Initial state distribution: \( p(x_0) \), Policy: \( p(u_t|x_t) \)
1: Init a segment \( s \) with \( p(x_0) = p(x_0) \) and weight \( w_0^s = 1 \)
2: for \( t = 0 \) to \( T \) do
3: for each \( s \) s.t. \( w_t^s > 0 \) do
4: Obtain \( p(x_t^s, u_t^s), p(i_{t+1}^s) \) \( \triangleright \) (6a), (6b), (6c)
5: for each \( p(i_{t+1}^s = i') \neq 0 \) do
6: Obtain \( p(x_t^{s'}, u_t^{s'}) \) \( \triangleright \) s' \( \rightarrow \) i'
7: if \( s = s' \) then
8: Get \( p(x_{t+1}^s) \) from \( p(x_t^{s'}, u_t^{s'}) \) \( \triangleright \) (6d)
9: Merge \( p(x_t^s, u_t^s) \) to \( p(x_{t+1}^s) \) \( \triangleright \) (7)
10: \( w_{t+1}^s = + p(i_{t+1}^s = i')w_t^s \)
11: if \( s \neq s' \) then \( \triangleright \) Init \( s' \) if required
12: Get \( p(x_{t+1}^s) \) from \( p(x_t^{s'}, u_t^{s'}) \) \( \triangleright \) (6e)
13: Merge \( p(x_t^{s'}) \) to \( p(x_{t+1}^s) \) \( \triangleright \) (7)
14: \( w_{t+1}^s = + p(i_{t+1}^s = i')w_t^s \)
of the next mode, \( p(i_{t+1}^s) \), at each time step \( t \) and for each active segment \( s \). We split \( p(x_t^s, u_t^s) \) into a number of \( p(X_t^{s'}, u_t^{s'}) \) based on all predicted switches \( s|s' \) (s to \( s' \)) as discussed earlier. There are two cases: \( s = s' \) (non-switching) and \( s \neq s' \) (switching). In the non-switching case, the state distribution is advanced using one of the dynamics models (line 9). The switching case follows a similar process but with one of the initialization models (line 12). In both cases, the result of the one-step prediction is merged (lines 9 and 13) with another one that may exist as the result of a previous iteration of the loop in line 3.

The merging process is implemented to handle probabilistic switching, i.e. simultaneously active segments involved in a multistep switch are maintained as weighted Gaussians by consolidating a segment’s existing quantity with the one split away from the other one.

The merging of the two close by Gaussians in lines 9 and 13 is done as follows. Let \( \mathcal{N}(\mu_1, \Sigma_1) \) and \( \mathcal{N}(\mu_2, \Sigma_2) \) represent the two Gaussians and \( w_1 \) and \( w_2 \) their respective weights. It can be shown that the mean and variance of the merged Gaussian distribution are:

\[
\mu = \bar{w}_1 \mu_1 + \bar{w}_2 \mu_2, \\
\Sigma = \bar{w}_1 \Sigma_1 + \bar{w}_2 \Sigma_2 + \bar{w}_1 \mu_1^T + \bar{w}_2 \mu_2^T - \mu \mu^T,
\]

where \( \bar{w}_1 \) and \( \bar{w}_2 \) are the normalized versions of \( w_1 \) and \( w_2 \), respectively. The weights of the segments are updated (lines 10 and 14) such that for any time \( t \), if more than one segment is active, the state distribution will be a valid mixture of Gaussian. Therefore, if the mode predictor continuously predicts multiple modes, a multimodal state distribution with each mode corresponding to a dynamics mode follows.

V. Experimental Results

We conducted two experiments to validate our method. First, a simulated block of mass was controlled to slide on a surface (1D) with abrupt changes in frictional properties. Here, we also demonstrate the multimodal state propagation aspect. Second, a 7-DOF robot arm (YuMi) was controlled to move such that it experienced unexpected contacts and sliding motions. In this case, our method was applied to a dynamics model with \( x_t \in \mathbb{R}^{14} \) and \( u_t \in \mathbb{R}^7 \).

As baselines for comparison, we used GP (SE kernel with ARD), manifold GP (mGP) in [8] and the uncertainty-aware ANN method in PETS [9](uANN). Both mGP and uANN encodes the epistemic and aleatoric uncertainties and, in principle, can model any nonlinear function. In all cases, we do not assume observation noise. In mGP, the parameters of an ANN feature space mapping and an SE kernel based on that feature space is jointly learned. The ANN based mapping is expected to provide enough flexibility to represent discontinuities. The uANN model encodes uncertainty using an ensemble of bootstrap models and propagates state distributions with a number of particles. It benefits from the high flexibility and scalability of ANNs. For all the three baselines, we sampled 50 trajectories up to the full episode lengths and followed it by density estimation (GMM) at each time step to make them comparable to our method. For both experiments, we report results with small and large datasets. This helps in analyzing how the methods scale with data size. Each dataset is split into train (\( D_{\text{train}} \)) and test (\( D_{\text{test}} \)) subsets. We report the average negative log likelihood (NLL) and root mean square error (RMSE) of individual time step predictions w.r.t the test trajectories. The RMSE is calculated for the most likely mode.

For DPGMM, the upper bound on the number of clusters \( K_{\text{max}} \) was set to 10 and 20 for the sliding mass and robot experiments, respectively. The concentration parameter \( \alpha \) was set to 0.1 for both cases. For the UT method, we used parameters, \( \alpha = 1, \kappa = 2, \) and \( \beta = 0 \). mGP was implemented on the GPflow framework [23]. For uANN, we used the published implementation of [9] and chose 5 ensembles for all cases. The TSin fault uncertainty propagation scheme was used (possible), in which particles are assigned to one of the ensembles permanently. All ANN network structures were fully connected layers.

A. The Sliding Mass Experiment

The sliding mass experiment is designed to have three dynamics modes, in which two are expected to be predicted simultaneously with some probabilities. A block of mass is controlled by a policy that pulls it to fixed destination along one direction. The policy is a simple linear Gaussian controller that outputs force based on the current state (position-velocity). Along the direction of motion, the surface imposes three dynamics modes (Fig. 3): free friction-less motion, free sliding motion halted due to sticking and slipping under kinetic friction.
Once it is stuck, the stochastic control policy may or may not overcome the friction force, thus resulting in a probabilistic scenario of possibly staying stuck and also possibly slipping. We expect discontinuities to appear at stick and also slip. The slip is implemented as an instantaneous velocity jump to 5 m/s. Stochasticity is introduced to the dynamics by injecting Gaussian noise to the state.

The experiment generated two datasets: $D_{15} = \{D_{15}^{\text{train}}, D_{15}^{\text{test}}\}$ consisting of $15 + 5$ trials, and $D_{40} = \{D_{40}^{\text{train}}, D_{40}^{\text{test}}\}$ consisting of $40 + 5$ trials. Each trial trajectory is simulated for $T = 75$ time steps with a stepping time of 0.05 seconds. Fig. 4a shows the result of $D_{40}$. We indicate different modes using different colors. We used the same policy and initial state distribution that were used for generating the data. All but one test trajectory eventually switched to the slip mode. The mean predictions are plotted by adjusting its transparency according to the probability weights for better clarity. The variance shown is the $2\sigma$ region for individual modes and is not adjusted for the weights. Also visible in the plot is the overlap indicating some slight reduction in the spread of NLL is noticeable. Our method occasionally underestimates the variances while all the baselines consistently overestimates them.

Fig. 4: Prediction results (sliding mass). Top (4a): Long-term prediction ($D_{40}$). Bottom (4b): Comparison of means ($D_{15}$)

### Table I: Sliding mass scores (avg. 10 restarts)

|      | $D_{15}$ | $D_{40}$ |
|------|----------|----------|
| NLL  | RMSE     | NLL      | RMSE    |
| GP   | 0.84 ± 3.04 | 1.77     | -        | -        |
| mGP  | 1.8 ± 2.94  | 3.0      | -        | -        |
| uANN | 0.92 ± 3.57 | 2.75     | 0.57 ± 3.3 | 3.53 |
| Ours | -2.32 ± 3.8 | 0.63     | -2.3 ± 3.45 | 0.76 |

Although both the means and variances are used for the score calculation. For the baselines, the density estimation of the sampled trajectories was done using DP-GMM ($K_{\text{max}} = 2$).

The robot was put in gravity compensated mode and torque controllers, and a closed-loop inverse kinematics scheme. Gaussian noise ($\mathcal{N}(0, 10^{-4})$) was added to the $X$ and $Y$ components of the reference trajectory as exploration noise. The robot was put in gravity compensated mode and torque commands were sent to the control interface.

Each trial consisted of 100 time steps sampled at 0.05 seconds interval. We also perturbed the base policy (controller) by altering the joint space proportional gains of all axes by $\pm 2\%$, $\pm 5\%$, and $\pm 10\%$. The experiment generated two datasets: $D_{15} = \{D_{15}^{\text{train}}, D_{15}^{\text{test}}\}$ and $D_{40} = \{D_{40}^{\text{train}}, D_{40}^{\text{test}}\}$.
where $\mathbb{D}_{15}^{\text{train}}$ contained 15 trials from the base policy and $\mathbb{D}_{40}^{\text{train}}$ had extra 5 trials each from all except the −10% case. The −10% case also had 5 trials and was designated as the test set $\mathbb{D}_{40}^{\text{test}}$ (= $\mathbb{D}_{15}^{\text{test}}$). By perturbing the policy, we aim to closely emulate a typical MBRL iteration in which the dataset for model learning would be a mixture of trials from slightly different policies. All aspects of our method is in the joint space, with the state variable comprising of joint positions and velocities. Our actual policy is a complex Cartesian space controller, but we use the generated joint space reference trajectory and the fixed PD gains as a simpler alternative. We use the resulting trajectory-centric policy that is associated with the test sets for long-term predictions.

In Fig. 6a, the results of $\mathbb{D}_{40}$ is presented in the Cartesian space (only translational position) for visual clarity. The transformation from the joint space was done by applying the UT method on forward kinematics. The plot shows only the most likely mode at each time step. It can be seen that the method sequentially switched through eight modes, each indicated with its own color. This reveals that the clustering step discovered eight clusters (potential over-modes, each indicated with its own color. This reveals that although it performed generally well up to relatively shorter horizon, as was used in [9], it ultimately went unstable by successively predicting larger values. We reduced this problem by resorting to a scheme of averaging predictions of all ensembles for each particle. Larger training data also alleviated this problem. While our method predicted fairly consistent variances, with occasional underestimations, all of the baselines produced overestimated variances. Our method has the best scores in all cases (Table II). mGP has better scores than uANN but the plot shows noisy mean predictions. As in the previous experiment, the GP outperformed the other baselines. Both GP and mGP were not feasible to train with $\mathbb{D}_{40}$. Our method showed slight improvement with $\mathbb{D}_{40}$, but the case with uANN was inconclusive. For $\mathbb{D}_{40}$, the training times were 53s and 923s for uANN and our method, respectively. This can be improved with parallelization.

### VI. DISCUSSIONS

Our method has high prediction accuracy and is able to handle discontinuities in a very data efficient manner thanks to its novelties such as the one-step-ahead mode predictor and the initialization model. It can be argued that ANN based methods would require much larger training data especially in the vicinity of discontinuities. A sign of data inefficiency in the case of uANN was the instability in long-term prediction. Both mGP and uANN are sensitive to network structure selection and also suffer from the problem of local optimum. The uANN method, however, scales better. While training uANN with minibatch gradient descent is usually fast, the parameter optimization of mGP with L-BFGS can be quite slow. We attribute the slightly better scores of the GP baseline over uANN and mGP to the above mentioned issues with ANN based models. Our method inherits data efficiency and minimum model selection from GP and has improved prediction accuracy by virtue of being a system of expert models. One aspect of accuracy is the variance prediction which we noticed as being significantly overestimated by all the baselines. The experiments clearly show that our method outputs reasonable variance estimations, even in the presence of discontinuities.

| $\mathbb{D}_{15}$ | NLL | RMSE | $\mathbb{D}_{40}$ | NLL | RMSE |
|---|---|---|---|---|---|
| GP | −4.6 ± 38 | 0.25 | - | - | - |
| mGP | −0.9 ± 12 | 1.22 | - | - | - |
| uANN | 5.1 ± 23 | 2 | - | - | - |
| Ours | −22.7 ± 26 | 0.17 | −24 ± 27 | 0.15 | - |

TABLE II: Contact motion scores (avg. 10 restarts)
Unlike PILCO, our particle based method does not offer the possibility to compute analytic gradients for use in policy search. A complete MBRL would be our future focus and a suitable strategy could be the black-box gradient-free approach in [24].

The clustering step relies on contact induced formation of clusters of data in the state space. However, not all mode transitions are accompanied by impacting contacts and i.i.d clustering such as DPGMM can easily identify multiple clusters within a mode as we have seen in the results. None of these is an issue because our focus has clearly been modeling and prediction involving discontinuities rather than precisely and uniquely identifying dynamics modes. The current clustering strategy could result in suboptimal solutions in the event of local optimum or sparse trajectories. Trajectory segmentation methods that take into account temporal correlation of data could be a future improvement.

Multimodality in long-term prediction is allowed to arise only due to simultaneous activation of modes and not from the propagation in a single mode. Such representations of discrete modes of operation could be exploited in policy search. Also, state distributions in parametric form removes the need for numerous trajectory samples for cost evaluation.

One of the biggest concerns with GP models is its scalability with training data \(O(n^3)\). Our divide and conquer strategy reduces training time to a great extent but could still improve. Since each output dimension of each dynamics model is represented by an independent GP, there is immense potential to improve the scalability of model learning with parallelization. This will be pursued in our future work.

VII. CONCLUSION

In this paper, we addressed the problem of predicting the state evolution of a contact-rich manipulation task using a learned dynamics model. In MBRL, such predictions are useful for evaluating a policy. Focusing on discontinuities as the main challenge, we adopted the strategy of a switching dynamics model and made progress on discontinuous state evolution. Our GP based method inherently encodes uncertainty due to limited data and it can output multimodal state distribution that can reflect discrete modes of operations. The main experiment, which closely reflected a typical MBRL iteration of an episodic manipulation task, showed that our method outperforms highly flexible ANN based methods. The need for achieving further scaling with data remains and a clear strategy has been suggested. Therefore, although limited to gradient-free settings, the proposed method is a promising option for model learning for contact-rich tasks.

REFERENCES

[1] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” The International Journal of Robotics Research, vol. 32, no. 11, pp. 1238–1274, 2013.

[2] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, “Gaussian processes for data-efficient learning in robotics and control,” IEEE transactions on pattern analysis and machine intelligence, vol. 37, no. 2, pp. 408–423, 2015.

[3] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, “Adaptive mixtures of local experts,” Neural computation, vol. 3, no. 1, pp. 79–87, 1991.

[4] S. Paolelli, A. L. Juloski, G. Ferrari-Trecate, and R. Vidal, “Identification of hybrid systems a tutorial,” European journal of control, vol. 13, no. 2-3, pp. 242–260, 2007.

[5] S. Linderman, M. Johnson, A. Miller, R. Adams, D. Blei, and L. Paninski, “Bayesian learning and inference in recurrent switching linear dynamical systems,” in Artificial Intelligence and Statistics, 2017, pp. 914–922.

[6] Y. Bengio and P. Frasconi, “An input output hmm architecture,” in Advances in neural information processing systems, 1995, pp. 427–434.

[7] D. Nguyen-Tuong and J. Peters, “Model learning for robot control: a survey,” Cognitive processing, vol. 12, no. 4, pp. 319–340, 2011.

[8] R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth, “Manifold gaussian processes for regression,” in Neural Networks (IJCNN), 2016 International Joint Conference on. IEEE, 2016, pp. 3338–3345.

[9] K. Chua, R. Calandra, R. McAllister, and S. Levine, “Deep reinforcement learning in a handful of trials using probabilistic dynamics models,” in Advances in Neural Information Processing Systems 31, 2018, pp. 4754–4765.

[10] R. Calandra, S. Ivaldi, M. P. Deisenroth, E. Rueckert, and J. Peters, “Learning inverse dynamics models with contacts,” in Robotics and Automation (ICRA), 2015 IEEE International Conference on. IEEE, 2015, pp. 3186–3191.

[11] M. Toussaint and S. Vijayakumar, “Learning discontinuities with products-of-sigoids for switching between local models,” in Proceedings of the 22nd international conference on Machine Learning. ACM, 2005, pp. 904–911.

[12] D. Nguyen-Tuong, J. R. Peters, and M. Seeger, “Local gaussian process regression for real time online model learning,” in Advances in Neural Information Processing Systems, 2009, pp. 1193–1200.

[13] C. D. McKinon and A. P. Schoellig. “Learning multimodal models for robot dynamics online with a mixture of gaussian process experts,” in Robotics and Automation (ICRA), 2017 IEEE International Conference on. IEEE, 2017, pp. 322–328.

[14] C. E. Rasmussen and Z. Ghahramani, “Infinite mixtures of gaussian process experts,” in Advances in neural information processing systems, 2002, pp. 881–888.

[15] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, “Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning,” in 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 7559–7566.

[16] I. Lenz, R. A. Knepper, and A. Saxena, “Deepmpc: Learning deep latent features for model predictive control,” in Robotics: Science and Systems, 2015.

[17] S. Levine, N. Wagener, and P. Abbeel, “Learning contact-rich manipulation skills with guided policy search,” in Robotics and Automation (ICRA), 2015 IEEE International Conference on. IEEE, 2015, pp. 156–163.

[18] C. E. Rasmussen, “The infinite gaussian mixture model,” in Advances in neural information processing systems, 2000, pp. 554–560.

[19] 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, vol. 12, pp. 2825–2830, 2011.

[20] J. Ko and D. Fox, “Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models,” Autonomous Robots, vol. 27, no. 1, pp. 75–90, 2009.

[21] S. J. Julier and J. K. Uhlmann, “New extension of the kalman filter to nonlinear systems,” in Signal processing, sensor fusion, and target recognition VI, vol. 3068. International Society for Optics and Photonics, 1997, pp. 182–194.

[22] M. P. Deisenroth, M. F. Huber, and U. D. Hanebeck, “Analytic moment-based gaussian process filtering,” in Proceedings of the 26th annual international conference on machine learning. ACM, 2009, pp. 225–232.

[23] D. G. Matthews, G. Alexander, M. Yan Der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman, “Gpflow: A gaussian process library using tensorflow,” The Journal of Machine Learning Research, vol. 18, no. 1, pp. 3186–3191, 2017.

[24] S. J. Julier and J. K. Uhlmann, “A new extension of the Kalman filter to nonlinear systems,” in Proceedings of the 1994 American Control Conference, vol. 3, pp. 182–186, 1994.

[25] B. Scutter, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” The international journal of robotics research, vol. 32, no. 11, pp. 1283–1267, 2013.

[26] D. I. Mandarin, H. A. Miller, R. Adams, D. Blei, and L. Paninski, “Bayesian learning and inference in recurrent switching linear dynamical systems,” in Artificial Intelligence and Statistics, 2017, pp. 914–922.

[27] K. Chua, R. Calandra, R. McAllister, and S. Levine, “Deep reinforcement learning in a handful of trials using probabilistic dynamics models,” in Advances in Neural Information Processing Systems 31, 2018, pp. 4754–4765.