Learning Non-Parametric Models in Real Time via Online Generalized Product of Experts

Connor Watson and Tania K. Morimoto

***Abstract*—In this work, we address the problem of online learning, where models must be continually updated from an incoming stream of data, while retaining past information. We develop an approach that is nonparametric, models uncertainty, and requires minimal hand-tuning. Our proposed algorithm, which we term online generalized product of experts (OGPoE), extends the powerful generalized product of experts (GPoE) framework to the online setting by leveraging methods for sparse, variational Gaussian process approximations, as well as nonparametric clustering. We devise a 1-D example learning problem to illustrate how our method works, and we verify that we achieve competitive results with other popular modeling approaches on a benchmark learning problem. Finally, we demonstrate how our algorithm can produce high accuracy predictions on a physical system, by learning the kinematics for a concentric tube robot, even when the robot is subject to changing, unknown loads.**

***Index Terms*—Model learning for control, modeling, control, and learning for soft robots, machine learning for robot control.**

I. Introduction

MANY real-world robotics problems involve phenomena such as friction and uncertainty that are difficult to model from first principles. These challenges have motivated the use of learning approaches that build models directly from data without requiring expert knowledge of the underlying system. While these learning-based approaches have achieved impressive results, many of the algorithms can only be performed offline, limiting their practical use.

Online learning algorithms seek to address this limitation by inferring models from data and adapting these models continually over time as new data arrives. In particular, nonparameteric online learning methods based on the popular Gaussian process (GP) statistical model are promising due to their many desirable properties, such as adaptable modeling complexity and prediction uncertainty quantification [1].

A. Related Works

Many approaches have been proposed to minimize the high computational cost of GP models, which scales cubically with the size of the dataset, N, during training [2]. Approximation strategies for GPs can typically be classified as either global or local methods. Global approximation strategies involve choosing $M \ll N$ points, called inducing or psuedo-inputs, that, when used for training, produce an approximate GP with minimal KL-divergence from a GP trained with all N samples. The inducing input locations may be sampled from the available data (based on, e.g., information gain [3], KL divergence [4], or Nyström sampling [5]), or considered as variables to be optimized for jointly with the kernel hyperparameters during training [6]. However, for online learning problems where the size of the dataset grows large over time, the quality of the GP approximation using a fixed number of inducing points may degrade [7] or be confined to a local region of the input space [8].

Local GP approximations improve scalability by first dividing the training dataset into smaller subsets (e.g. by k-means [9], randomly [10], or by feature space distance from existing local models [11]) and then training a local GP model on each subset of data. Prediction on new inputs can be done with the local model nearest the input (i.e. naive local experts (NLE) [2]) or by taking a weighted combination of predictions from all local models (i.e. mixture of experts (MoE) [12] or product of experts (PoE) [13]). Adapting this local GP approximation strategy to the online setting is difficult because most of the effective data partitioning and prediction combination methods that have been used to date, require some form of offline computation.

Recent works have proposed to combine both local and global GP approximations to leverage the benefits of each approach [14], [15]. Training data is divided between multiple, local models, where each local model is itself a sparse GP approximation. Although these approaches substantially reduce training times and effectively scale the complexity of the models to the observed data, there remain limitations in their prediction strategies. For instance, [15] uses an NLE strategy, which is known to give artificially discontinuous predictions on the boundaries of the subregions between local models [2]. The prediction strategy in [14], on the other hand, uses an MoE approach to prediction, but does not learn the weighting function jointly with the local model parameters, which typically leads to suboptimal predictions [10].

To address remaining limitations in using GPs for online learning, we propose to adapt the generalized product of experts (GPoE) framework for multiple model prediction [10] to the online setting. GPoE is a method for combining the predictions from multiple local models in a way that does not require joint training, allowing each local model to be learned independently. Despite this independent training, the aggregate GPoE prediction has been shown to retain important properties of multiple model prediction, such as rejection of poor predictions from local models, smooth interpolation between local models, and

Manuscript received 23 February 2022; accepted 28 June 2022. Date of publication 14 July 2022; date of current version 25 July 2022. This letter was recommended for publication by Associate Editor C. Della Santina and Editor C. Gosselin upon evaluation of the reviewers' comments. This work was supported by the National Science Foundation under Grant 1935329. *(Corresponding author: Connor Watson.)*

Connor Watson is with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: cmwatson@eng.ucsd.edu).

Tania K. Morimoto is with the Department of Mechanical and Aerospace Engineering and the Department of Surgery, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: tkmorimoto@eng.ucsd.edu).

Digital Object Identifier 10.1109/LRA.2022.3190809

the ability to precisely combine many uncertain local model predictions [10].

B. Contributions

The contributions of this paper are as follows. First, we present a new algorithm, which we call online generalized product of experts (OGPoE), that extends GPoE to the online learning setting. This algorithm offers a combination of benefits unavailable from other online learning algorithms, including the simultaneous online optimization of local model parameters, a clear and interpretable data partitioning strategy, and principled fusion of multiple model predictions without the need for joint training. We evaluate our proposed algorithm in simulation and show that it compares favorably to other popular online and offline modeling approaches on a well-cited benchmark dataset. Finally, we demonstrate how OGPoE can be used for online learning tasks that involve physical hardware and dynamic environments. Specifically, we demonstrate online learning of the kinematics of a concentric tube robot — both in free space and in the presence of dynamic environmental constraints — which is a first for this type of robot. These experiments not only demonstrate the modeling accuracy that OGPoE can achieve, but also its versatility in the types of problems it applies to, making it a powerful tool for online learning.

II. Background

In this section, we review GP fundamentals, along with key works that are leveraged and adapted for OGPoE.

A. Streaming Sparse Gaussian Process Approximations

A standard GP regression model assumes that input-output data pairs, $\{\mathbf{x}, y\}$, can be described by the expression $y = f(\mathbf{x}) + \epsilon$, where f is an unknown function corrupted by Gaussian observation noise $\epsilon \sim \mathcal{N}\left(0, \sigma_y^2\right)$ [7]. A GP prior over the unknown function, $f \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot|\theta))$, is specified by a mean function, $m(\cdot)$, and a covariance function, $k(\cdot,\cdot)$, that depends on hyperparameters θ. As is typically done, we adopt a zero-mean function for notational convenience in the remainder of this section. The choice of covariance (or kernel) function encodes prior assumptions about the spatial correlations between the inputs to the unknown function, f, and may be informed by prior knowledge. We employ the popular and interpretable squared exponential kernel with automatic relevance determination,

$$k\left(\mathbf{x}, \mathbf{x}'\right) = \sigma_f^2 \exp\left(-\frac{1}{2}\left(\mathbf{x}-\mathbf{x}'\right)^\top \mathbf{W}^{-1}\left(\mathbf{x}-\mathbf{x}'\right)\right), \quad (1)$$

which depends upon the hyperparameters that capture the signal variance (σ_f^2) and the length-scale along each dimension of the input ($\mathbf{W} = \text{diag}(l_1^2, \ldots, l_d^2)$).

Given a collection of input-output data pairs, the assumption of Gaussian observation noise, and the GP prior over f, it is possible to compute the mean and variance of the posterior process in closed form at a new input, $\mathbf{x}_*$, as

$$\begin{aligned} m_*(\mathbf{x}_*) &= k(\mathbf{x}_*, \mathbf{X})\left(\sigma_y^2\mathbf{I} + k(\mathbf{X},\mathbf{X})\right)^{-1}\mathbf{y} \\ V_*(\mathbf{x}_*) &= k(\mathbf{x}_*,\mathbf{x}_*) - k(\mathbf{x}_*,\mathbf{X})\left(\sigma_y^2\mathbf{I} + k(\mathbf{X},\mathbf{X})\right)^{-1}k(\mathbf{X},\mathbf{x}_*) \end{aligned} \quad (2)$$

The kernel matrix, $k(\mathbf{X},\mathbf{X})$, consists of kernel function evaluations between all N datapoints, and must be inverted for inference. As N approaches $\sim 10^4$, this computation becomes intractable, motivating methods for choosing $M \ll N$ inducing inputs to approximate the posterior and retain tractability [6]. The vast majority of these approximation methods, however, can only be applied when all training data is available at once, which is not the case in the online setting where data is revealed incrementally over time.

To address this problem, the authors in [7] pose an optimization problem that *recursively* updates the locations of M inducing points, $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_M\}$, as well as the kernel hyperparameters, $\theta = \{\sigma_y^2, \sigma_f^2, \mathbf{W}\}$, of a sparse GP model as data is processed online. Specifically, they show that given some current posterior GP approximation, $\mathcal{GP}_{\text{old}}$, with hyperparameters, θ_{old}, and inducing input locations, $\mathbf{Z}_a$, that is assigned new data, $\mathcal{D}_{\text{new}} = \{\mathbf{x}_i, y_i\}_{i=1}^{Nnew}$, then a quantity called the online variational free energy ($\mathcal{F}_{\text{OVFE}}$) can be maximized to find values for new hyperparameters and inducing input locations, $\{\theta_{\text{new}}, \mathbf{Z}_b\}$. The new hyperparameters and inducing input locations are then used to form a new posterior approximation, $\mathcal{GP}_{\text{new}}$, which, in turn, can be used for prediction [7].

B. Generalized Product of Experts

The PoE method for ensemble prediction [13] attempts to model a target distribution as the product of multiple densities, each of which is given by a single expert. This method is sensitive to errors in the predicted variance of the local experts, in that erroneously low-variance predictions from some local models will bias the aggregate mean and lead to overconfident predictions [10]. To suppress the influence of poor experts on the overall model, GPoE [10] introduces a weight to each expert's prediction,

$$p(y_*|\mathcal{D}, \mathbf{x}_*) = \frac{1}{Z}\prod_{i=1}^{M} p_i^{\beta_i(\mathbf{x}_*)}(y_*|\mathcal{D}_i, \mathbf{x}_*), \quad (3)$$

where Z is a normalizer, $\mathcal{D}_i$ is the training data seen by the ith model, $\mathbf{x}_*$ is a predictive input, and y_* is an output. The input-dependent weight function, $\beta_i(\mathbf{x}_*) = \frac{1}{2}(\log\sigma_{i,**}^2 - \log V_i(\mathbf{x}_*))$ is the differential entropy between the expert's prior and posterior. When the entropy change for an expert is zero at $\mathbf{x}_*$, then that expert's prediction does not contribute to the combined prediction because it does not have any training information relevant to this new input. Using this weight and (3), the aggregate mean (m_{GPoE}) and variance (V_{GPoE}) of the ensemble of experts can be computed in closed form [10]. This simple strategy for weighting predictions still requires no joint training and alleviates the PoE problem of being unable to reject poor experts. Despite its simplicity and minimal computational requirements, GPoE achieves excellent results on multiple benchmarks [10].

C. DP-Means

DP-means [16], [17] is a nonparametric clustering technique that allows data to be partitioned for local model training *without* specifying the number of clusters *a priori*. We consider a set of streaming data, $\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, where $\mathbf{x}_t \in \mathbb{R}^d$ is the data observed at time t. Each observation is assumed to belong to one of K clusters and is labeled $\{\zeta_1, \ldots, \zeta_t\}$, with $\zeta_t \in \{1, \ldots, K\}$. A prior over the number of clusters, K, and the cluster assignments, $\zeta_{1:t}$, specified by a Chinese Restaurant Process (CRP) [18] with

hyperparameter α, is assumed. The first observation, $\mathbf{x}_1$, is assigned to the first cluster, $\zeta_1 = 1$, and the tth observation is assigned either to existing cluster k, with probability proportional to the number of data already in that cluster, or to a new cluster, with probability proportional to α. If the data in each cluster is assumed to be Gaussian with cluster-specific means, $\boldsymbol{\mu}_k$, and a shared variance, $\sigma^2\mathbf{I}_d$, then the likelihood of the data, $\mathbf{x}_{1:t}$, given clustering $\zeta_{1:t}$ and means $\boldsymbol{\mu}_{1:k}$, is

$$p(\mathbf{x}|\zeta, \boldsymbol{\mu}) = \prod_{k=1}^{K}\prod_{t=1}^{T}\mathcal{N}\left(\mathbf{x}_t|\boldsymbol{\mu}_k, \sigma^2\mathbf{I}_d\right). \tag{4}$$

Further assuming a zero-mean Gaussian prior on the cluster means themselves, $\boldsymbol{\mu}_k \sim \mathcal{N}(0, \rho^2\mathbf{I}_d)$, Bayes theorem gives the posterior distribution over the clustering,

$$p(\zeta, \boldsymbol{\mu}|\mathbf{x}) \propto p(\mathbf{x}|\zeta, \boldsymbol{\mu})p(\zeta)p(\boldsymbol{\mu}). \tag{5}$$

The *maximum a posteriori* (MAP) estimate for the clustering and the cluster means is found by maximizing this posterior distribution or, equivalently, minimizing the negative log joint likelihood

$$\arg\max_{K,\zeta,\boldsymbol{\mu}} p(\zeta, \boldsymbol{\mu}|\mathbf{x}) \propto \arg\min_{K,\zeta,\boldsymbol{\mu}} -\log p(\mathbf{x}, \zeta, \boldsymbol{\mu}). \tag{6}$$

Typically, computing the MAP estimate is quite difficult, but by choosing the hyperparameter $\alpha = \exp(-\lambda/2\sigma^2)$ and taking the small variance asymptotic limit as $\sigma^2 \to 0$, the optimization reduces to

$$\arg\min_{K,\zeta,\boldsymbol{\mu}} \sum_{k=1}^{K}\sum_{t=1}^{T}\|\mathbf{x}_t - \boldsymbol{\mu}_k\|^2 + \lambda K. \tag{7}$$

Note this objective function takes the same form as that of the k-means algorithm plus a penalty term for adding new clusters. It is solved straightforwardly by the DP-means algorithm, which iteratively assigns data to the cluster with the nearest mean unless the nearest cluster mean exceeds a threshold distance, controlled by the user-set value λ, upon which a new cluster is generated. Although more complicated extensions to the DP-Means algorithm exist (e.g. [16], [19]), DP-Means requires few user-set parameters and often leads to satisfactory results.

III. Online Generalized Product of Experts

In this section, we present Online Generalized Product of Experts (OGPoE), which consists of a partitioning stage, a training stage, and a prediction stage. We present the algorithm design and implementation, and we discuss how the specific combination of algorithms gives OGPoE several advantages compared to the state-of-the-art in online learning.

A. Algorithm Design

To overcome current limitations in using GPs for online learning, each component of the OGPoE algorithm must be carefully selected. First, the training component should provide a principled method for updating the model inducing point locations and hyperparameters. Previous works either update these values separately [20], require that the hyperparameters be known *a priori* [11], or implicitly assume that the training data is independently, identically distributed (iid) [21]. We select to use streaming sparse GP approximations in order to simultaneously update both the inducing point locations and hyperparameters even when the training data is generated from a time series and is not iid.

Second, in order to grow the model complexity as the amount of training data increases, the partitioning strategy must be simple, easily tune-able, and interpretable, in order to divide data amongst an ensemble of local models. Previous works have used somewhat opaque online partitioning strategies [11], [14], [15] that make it difficult to predict when a new model will be formed, and consequently more challenging to tune. We select to use the DP-means algorithm, which can be analysed through the lens of small variance asmpytotics [17] to better understand its inherent assumptions and convergence. This simple strategy is regulated by a single user-set parameter that is straightforward to tune.

Lastly, the prediction strategy must be able to fuse predictions from multiple local models, online, without imposing restrictive assumptions on how the models are trained. Many prediction strategies, however, assume all local models share the same hyperparameters ([11], [22]). We therefore select to use the GPoE framework, which allows for each of the individual experts to be trained independently and the aggregate prediction to better capture non-stationary features in the dataset [2]. Further, GPoE does not require the data used for training each expert to be local, and produces good aggregate predictions even when the individual experts are trained on random subsets of data that may significantly overlap [10], as could be the case when using a sparse GP approximation. Together, these features allow OGPoE to build models online that describe data exhibiting complex phenomena such as non-stationarity and heteroscedastic noise.

B. Implementation

Implementation of OGPoE consists of an initialization phase, followed by online learning and prediction (Fig. 1).

Initialization: We assume access to $N_{\text{init}} > M$ samples which are used to initialize our model. When learning a kinematics or dynamics model, for example, these samples might be acquired by sending random, low-magnitude signals to the robot joints. We assign all initial data to a single cluster, which is represented by its mean, $\boldsymbol{\mu}_1 = \frac{1}{N_{\text{init}}}\sum_{i=1}^{N_{\text{init}}}\mathbf{x}_i$. A sparse GP model is then instantiated as in [6] by solving an optimzation problem to determine initial inducing input and hyperparameter values.

Online Training: Data arrives sequentially in small batches of N_{new} samples for processing. Each data point is assigned to either an existing cluster or to a new cluster by solving the optimization problem in (7). Once all data in a batch has been assigned, existing cluster means are updated with the data assigned to that cluster. The assigned data is then used to update the sparse GP approximation associated with each cluster by optimizing the online variational free energy objective as described in Section II-A. If a new cluster is generated, the sparse GP approximation of the nearest existing cluster is used for this optimization. This process can be repeated as long as there are batches of data to process.

For all model training, we empirically find it helpful to add an inequality constraint to the online VFE optimization problem that constrains each of the lengthscales of the squared exponential kernel to not be greater than twice the legnthscale parameter used in the clustering algorithm, $l_i \leq 2\lambda\ \forall i \in \{1, ..d\}$. Intuitively, the kernel function lengthscales, which dictate the correlations between datapoints, should not exceed the diameter of the cluster

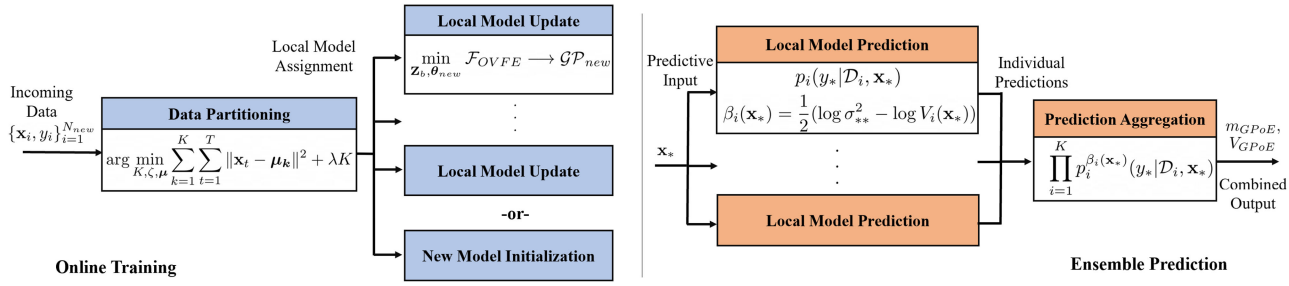

Fig. 1. Flowcharts depicting online learning with incoming data and the GPoE framework for ensemble prediction. Online learning assumes that data arrives sequentially in small batches of N_{new} samples that are used either to update an existing local model, or to generate a new local model. Ensemble prediction returns an aggregate predictive mean and variance, m_{GPoE}, V_{GPoE}, for an input $\mathbf{x}_*$.

from which they originate. Note that this constraint does not require the individual kernel lengthscales to be equal to one another, but merely enforces an upper bound.

Prediction: In order to combine predictions from multiple local models on a new input, we use the GPoE framework described in Section II-B. One disadvantage is that for test points far from any training data, the difference between the prior and posterior variance for all experts is nearly zero [9], [22], causing all of the weights to go to zero and the predictive variance of the aggregate model to explode (i.e. as $\beta_i \to 0$, $V_{\text{GPoE}} \to \infty$). Although this is only a minor concern for most online learning problems since predictions are made close to the training data which is revealed incrementally over time, we choose to mitigate this issue by simply normalizing the weights such that $\sum_{i=1}^{K} \beta_i = 1$ for prediction.

User Set Parameters: OGPoE requires little tuning to apply to new problems. We provide the following suggestions for the only two parameters, M and λ, that must be set by the user. It is worth noting that these are distinct from the kernel hyperparameters, θ, which are determined automatically from the data by optimization. M is the number of inducing inputs per sparse GP model that will be used for training. The question of how to choose M, such that $M \ll N$ points are guaranteed to approximate a GP model with tight KL divergence bounds is still an area of active research [23]. Practically, we find that setting the value for M presents a tradeoff between increased computation time for training and better modeling accuracy. Therefore, we advise choosing M to be as large as possible, while still maintaining a model update rate that is suitable for the particular application. The other parameter that must be set by the user is the cluster lengthscale, λ, used in the DP-means algorithm for data partitioning. Because of the similarity between DP-means and k-means, if a subset of data is available for offline computation, we find that clustering this data with k-means and then setting λ to be the average radius of the resultant clusters, is an effective initial guess for this parameter that may be further tuned as needed. Finally, although the batch size of training data is typically not controlled by the user since data is streaming at a fixed rate and processed as fast as possible, N_{new} can be set for simulated experiments. In this case, larger values of N_{new} increase the training time per batch of data, but can lead to better fits ([7]).

IV. Simulated Evaluation

We assess the performance of OGPoE through numerical experimentation. We present a simple learning problem to demonstrate the effectiveness of OGPoE for multiple model prediction, and we compare its performance to other popular modeling approaches on the SARCOS arm dataset.

A. Toy Example

We consider the problem of learning an approximation to the 1-D function $y = \text{sinc}(x) + \varepsilon$, with $\varepsilon \sim \mathcal{N}(0, 0.04)$. We generate 500 samples from this function for training, which we process sequentially from $x = -4$ to $x = 4$ (excluding $x \in [-0.5, 0]$) in batches of $N_{\text{new}} = 20$ samples, in order to simulate online training. For testing, we compare against the noise-free function $\bar{y} = \text{sinc}(x)$ for $x \in [-5, 5]$. With $\lambda = 0.5$ and $M = 10$, online training produces 4 local models with inducing point locations and individual predictions shown in Fig. 2 a. No one local model learns the entire latent function — with an RMSE of 0.091 for the best local model — necessitating an ensemble prediction strategy. There is overlap between the models, which can be attributed to the fact that each new local model is initialized using the nearest, existing local model, and the optimization used to update the locations of the local model inducing points can find values outside the region of the training data [7], [14]. Despite the overlap, Fig. 2 b shows that OGPoE can effectively combine local model predictions and capture the underlying function (0.059 RMSE), because it does not assume that the individual models cover independent regions of the input space. Instead, OGPoE effectively blends contributions from proximal models in an aggregate prediction.

Other ensemble prediction strategies are shown in Fig. 2 c and 2 d. NLE, for example, uses the predictive mean and covariance of the local model nearest the predictive input (Fig. 2 c). Consequently, its prediction contains artificial discontinuities where there is a transition between the local models used, despite the smoothness of the underlying function (0.085 RMSE). Heuristic MoE (Fig. 2 d) trains an additional meta GP model used to weight the predictive mean from each local model [11], [14]. Because this method combines the predictions of multiple models, it avoids the discontinuities associated with the NLE method but at the cost of additional training time. Additionally, because the local model weights are exponential in the distance between the model center and the predictive input, most of the weight for prediction is usually concentrated on a single model, producing a predictive output that is not substantially better than prediction with NLE (0.084 RMSE).

B. SARCOS Arm Benchmark

In this section, we study the effects of varying OGPoE's user-set parameters, λ and M, and compare our proposed algorithm

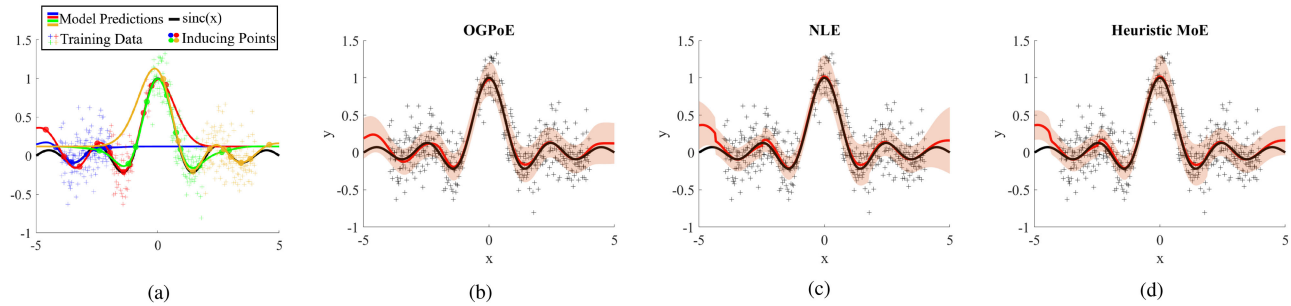

Fig. 2. (a) Sparse, local GP models are learned using data, '+,' drawn from a noisy, unknown function as detailed in Section IV-A. The predictions from these local models are combined using (b) OGPoE, (c) NLE, and (d) a heuristic MoE approach. The predictive mean of each aggregation method is shown in red along with the one standard deviation confidence bounds in orange.

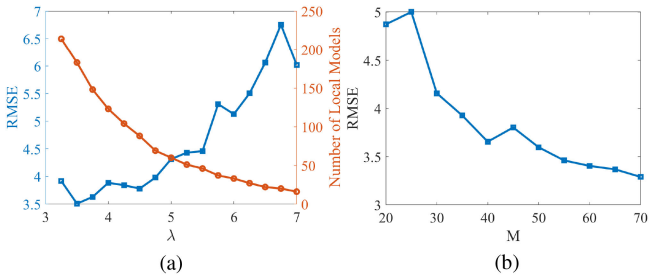

Fig. 3. Plots depicting training results on the SARCOS arm benchmark for different values of the user-set parameters (a) λ and (b) M.

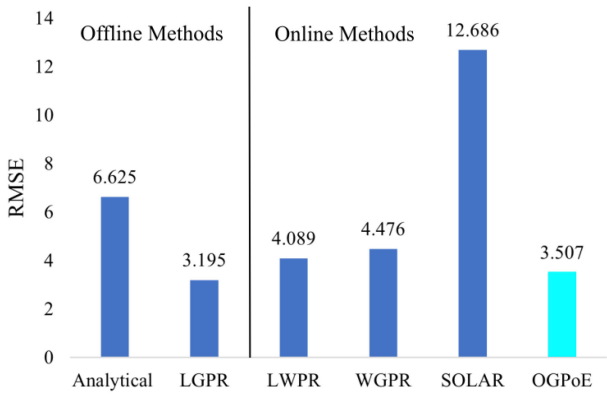

Fig. 4. SARCOS benchmark results demonstrating OGPoE achieves excellent performance relative to the state of the art in online learning methods. We report the results of SOLAR, WGPR, and LGPR from [15] and the analytical model and LWPR from [24].

to other modeling approaches, using the SARCOS arm dataset.[1] This publicly available dataset is frequently used to benchmark the performance of offline [1], [9], [11] and online [14], [15], [24] modeling approaches, making it a suitable standard to compare against without having to reproduce the results of many previous works. The objective is to learn an inverse dynamics mapping from a 21 dimensional input joint space to the first of 7 output joint torques, using a dataset that consists of 48,933 samples (44,484 for training and 4,449 for validation) [1], [9], [15].

First we assess the effect of the clustering lengthscale parameter, λ. We set $M = 50$ and process data sequentially in batches of $N_{new} = 100$ samples for training, as is done in [15], and then evaluate the prediction of the learned model on the validation dataset. We repeat this simulation for $\lambda \in [3.25, 7]$ (Fig. 3 a). As expected, the performance degrades as lambda is increased since the number of local models generated, and consequently the total number of inducing points used to capture the data, decrease. Next, to assess the effect of M on learning performance, we fix λ to the best value from Fig. 3 a and repeat the experiment, this time varying M from 20 to 70. The results, shown in Fig. 3 b, depict how performance improves with increasing M, although with diminishing returns as M becomes large.

Finally, we report the results of other popular modeling approaches on the SARCOS dataset in Fig. 4. The value of M used for OGPoE was set to 50 in order directly compare against the methods studied in [15], while λ was set to 3.5, which was the best result from the experiment shown in Fig. 3 a. Looking at the performance of LGPR [11], which requires offline optimization of the hyperparameters, we would expect it to significantly outperform OGPoE, where the hyperparameters are optimized online. However, we note that we achieve nearly comparable results, with a RMSE of 3.507 compared to 3.195 reported in [15].

The online methods we compare to are the well-known LWPR [24], as well as SOLAR-GP [14] and WGPR [15], which are the most similar algorithms to OGPoE. Our results show that we outperform these methods on this benchmark, even for a wide range of values of λ (Fig. 3 a). Because SOLAR and WGPR leverage the same sparse GP approximation strategy as OGPoE, the difference in performance can likely be attributed to differences in how data is partitioned between local models and how combined predictions are generated. OGPoE's data partitioning scheme differs in that data assignment is independent of the parameters used to define the local models, which change over time. The prediction framework employed by OGPoE more effectively combines information from multiple local models, leading to improved aggregate prediction as discussed in Section IV-A.

V. Hardware Evaluation

In this section, we use OGPoE to learn the forward kinematics of a concentric tube robot (CTR) from streaming data. CTRs are a class of continuum robot made up of precurved, flexible, telescoping tubes that when translated and rotated relative to one another, interact in bending and torsion to reach different equilibrium configurations [25], [26]. Mechanical models based on

[1]Dataset available at: http://www.gaussianprocess.org/gpml/data/

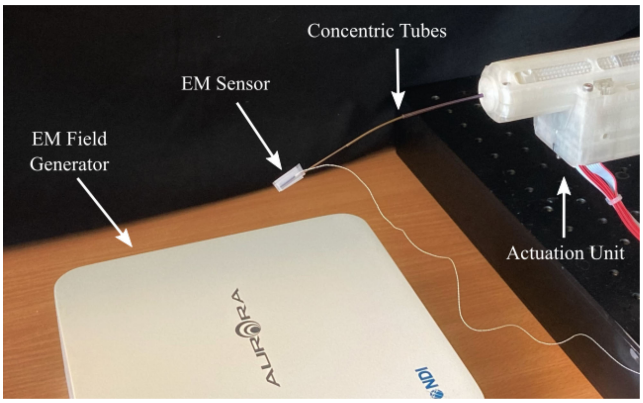

Fig. 5. The experimental setup consists of a 3-tube CTR and associated actuation unit from [29]. An EM sensor is attached to the CTR tip in order to track its position.

the theory of Cosserat Rods [27] have been developed to describe their forward kinematics, where the tip position is computed as a function of the rotational and translational positions of each of the tubes [28]. Typically, these models assume that all tubes share a common centerline and ignore the effects of friction and clearance between the tubes. While these assumptions lead to a tractable set of equations for the kinematics, they also contribute to modeling inaccuracies. We approach learning kinematics for these robots in two ways: one in which the forward kinematics must be learned completely from scratch, and another in which we augment the learned model with a mean function derived from first principles.

A. Experimental Setup

The 3-tube CTR and associated actuation unit from [29] were used in this experiment (Fig. 5). An Aurora electromagnetic (EM) tracking system (NDI) is used to measure the 3-D position of an EM sensor (0.92 mm outer diameter) attached to the tip of the robot. Joint and EM sensor positions are sent to the OGPoE algorithm to be used for training via the Robot Operating System (ROS) [30], which is used to enable parallel processes for prediction and training [14]. An open-loop joint space trajectory is also published over ROS to set the joint positions and to predict the robot tip position from the learned model.

For our hybrid learning approach, which involves augmenting the learned prediction with an analytical model, we follow the model derivation of [28]. This approach amounts to solving a set of ordinary differential equations subject to boundary conditions (BCs) imposed by the joint values of the tubes. For our CTR, these equations have a unique solution for every set of BCs dictated by the robot's joint configuration, defining a unique mapping that we use as our analytical kinematic model. Because this mapping is deterministic, we may incorporate it into our learning problem by adopting a zero-mean GP prior over the *difference* between the tip position predicted by this model and the true tip position observations measured by the EM sensor for learning. For prediction, we combine the output of the analytical model with this learned error function. To save computation time, we follow the example of [24] and [14] by using the same learned model to predict each component of the tip position.

B. Online Learning of Robot Kinematics in Free Space

In this experiment, the CTR is commanded to follow an open-loop joint-space trajectory consisting of 600 points that was designed offline to cover a significant region of the robot workspace. At each timestep, we predict the robot tip position for the next joint configuration using one of three models — the Cosserat-rod-based kinematics model, the OGPoE model, or the hybrid model. To initialize the learned models, we sample random joint configurations in close joint space proximity to the robot's home configuration, which results in small tip motions. Specifically, we draw 50 samples for each joint from $\mathcal{U}(q_0 - 2.5\,\text{mm}, q_0 + 2.5\,\text{mm})$ for translational joints and from $\mathcal{U}(q_0 - 5^\circ, q_0 + 5^\circ)$ for rotational joints, where q_0 is the home position of the joint and $\mathcal{U}(a, b)$ is a uniform random distribution with lower limit a and upper limit b. These initial models are updated online according to the OGPoE framework as data arrives over the course of the trajectory and used to predict on new inputs just outside the distribution of training data. When the robot reaches the end of the trajectory, it returns to the starting position and then follows the same trajectory again to assess the behavior of the learned models when re-entering previously visited regions of the state space. For both the hybrid and purely learned OGPoE models, we set $\lambda = 3.75$ and $M = 25$, which we found to be acceptable after minimal tuning.

The prediction results in Fig. 6 show that there is significant error in the analytical model predictions (4.8 mm $\pm$ 1.52 mm) primarily in predicting the x and y components of the tip position. This error is expected for these models [28], [31], and a large part is likely due to ignoring clearance between the tubes of the robot, which causes the true shape to be straighter than the Cosserat rod model predicts [32]. Consequently, the path traced by the robot tip has a smaller radius than the one predicted by the model (Fig. 6 a). Both of the learned models, on the other hand, achieve much lower mean absolute error for predicting the tip position — 0.79 mm $\pm$ 0.48 mm and 0.58 mm $\pm$ 0.4 mm for the purely learned and hybrid models, respectively — demonstrating the feasibility of using OGPoE in an online learning scenario. The prediction error of the learned models decreases slightly between the two passes through the trajectory, while the number of local models stays constant, indicating that these learned models retain information about regions of the state space they have already visited. While both approaches achieve high accuracy predictions, the hybrid approach exhibits a slightly smaller average error due to incorporation of a good, physics-based, analytical mean function.

The success of OGPoE can be attributed to the fact that it updates a model online with data from the trajectory that is being followed, allowing for good predictions on new, nearby inputs. On average, the model update rate is 0.3 s and the prediction rate is 106 Hz, which is sufficient for surgical procedures where the robot speed will not exceed a few cm/s [33]. The benchtop setup can also be substantially improved with specialized hardware and software.

C. Online Learning of Robot Kinematics With Dynamic Loading Conditions

Next we demonstrate the ability of OGPoE to learn the kinematics of a CTR when the robot is subjected to an unknown and changing environmental constraint. This ability is particularly important for CTRs, since many of their intended applications

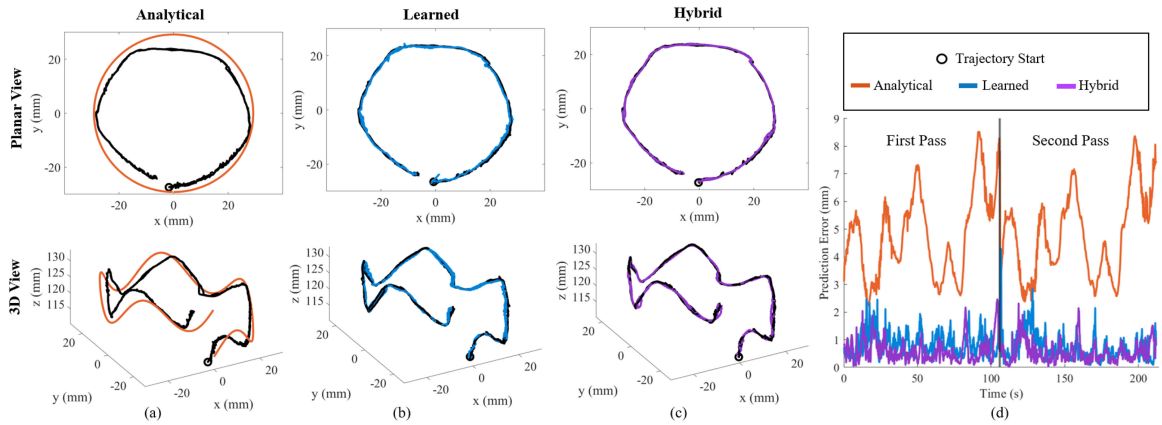

Fig. 6. Results of the online kinematics learning task using (a) an analytical model, (b) a model learned with OGPoE, and (c) a hybrid of the two approaches, where OGPoE is used to learn the error in the analytical model. The plots on the left show the predicted and measured tip positions from two different angles for each of the models used, and the plot on the right shows the prediction error over time for all three models.

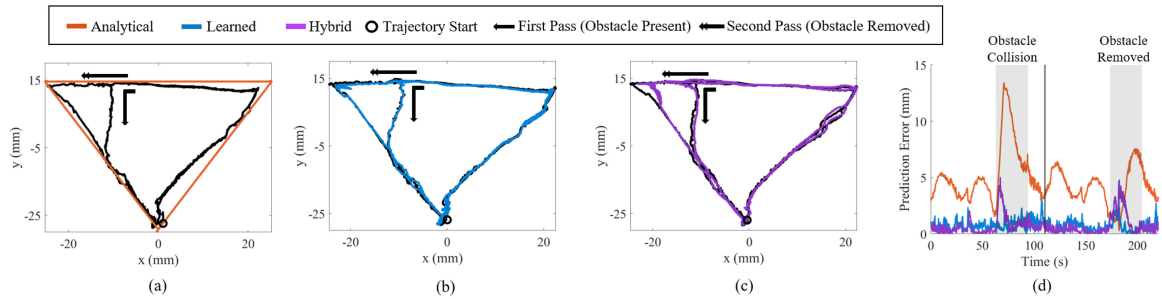

Fig. 7. Predicted tip position of the CTR following a planar, triangle-shaped path using an (a) analytical (b) learned and (c) hybrid model as well as (d) the prediction errors over time. On the first pass through the trajectory, an obstacle blocks the robots path, causing a large deviation from the analytical model. On the second pass through, the obstacle is removed.

involve navigation through dynamic environments, yet there is a notable lack of proposed solutions in the literature due to the difficulty of the problem.

We initialize the robot as in Section V-B and command it to follow a new open-loop trajectory, twice. This time however, an un-modeled obstacle is placed in the path of the robot, such that the robot tip is significantly deflected when it comes into contact with the obstacle. Before the robot traverses the trajectory a second time, the obstacle is removed, resulting in different tip position measurements that conflict with the data gathered on the first pass. Fig. 7 shows the robot tip position predicted by the analytical, learned, and hybrid models, along with the associated prediction errors.

As seen in Fig. 7(a), the presence of the obstacle causes large errors in the analytical model prediction (4.73 ± 2.23 mm average, 13.42 mm maximum), whereas the OGPoE-based methods achieve much more accurate results (learning: 0.75 ± 0.49 mm average, 4.67 mm maximum; hybrid: 0.72 ± 0.84 mm average, 5.00 mm maximum). It is also interesting to note that the error in the hybrid modeling approach has a small, transient increase when the robot encounters the obstacle (presumably due to the inaccuracy of the analytical model) and again when the robot traverses the region of the state space where the obstacle has been removed (where the previously learned deviation from the analytical model needs to be un-learned). Nonetheless, the low average prediction error of both OGPoE-based methods demonstrates the applicability of this method for learning CTR kinematics under these challenging conditions.

VI. Conclusions and Future Work

In conclusion, we presented OGPoE and illustrated its effectiveness starting with a simple 1-D example. We verified that OGPoE achieves excellent results on the SARCOS arm benchmark relative to the state of the art, making it a potentially useful tool for complex online learning problems. Finally, we demonstrated the ability of OGPoE to learn the kinematics for a continuum robot in an online setting, both with and without prior model information and even in the presence of changing, unknown loading conditions.

There are several directions for future work that can further expand the applicability of OGPoE. For example, the development of a method to bound the prediction error of the aggregate model learned by OGPoE can enable its use in online learning applications that require guaranteed safety, as have been recently studied using standard GPs with known hyperparameters [34]. Additionally, dimensionality reduction techniques could potentially be used with OGPoE to do effective online learning for problems with very high dimensional inputs. Finally, OGPoE could be applied to many other robotics problems, including learning and adapting a model for use with inverse kinematics methods to enable real-time control in unknown environments.

REFERENCES

[1] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Berlin, Germany: Springer, 2003, pp. 63–71.

[2] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When Gaussian process meets Big Data: A review of scalable GPs," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4405–4423, Nov. 2020.

[3] M. W. Seeger, C. K. Williams, and N. D. Lawrence, "Fast forward selection to speed up sparse Gaussian process regression," in *Proc. Int. Workshop Artif. Intell. Statist.*, 2003, pp. 254–261.

[4] G. Chowdhary, H. A. Kingravi, J. P. How, and P. A. Vela, "Bayesian nonparametric adaptive control using Gaussian processes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 3, pp. 537–550, Mar. 2015.

[5] C. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 661–667.

[6] M. Titsias, "Variational learning of inducing variables in sparse Gaussian processes," in *Proc. 12th Int. Conf. Artif. Intell. Statist.*, 2009, pp. 567–574.

[7] T. D. Bui, C. Nguyen, and R. E. Turner, "Streaming sparse Gaussian process approximations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3299–3307.

[8] F. Meier and S. Schaal, "Drifting Gaussian processes with varying neighborhood sizes for online model learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 264–269.

[9] H. Liu, J. Cai, Y. Wang, and Y. S. Ong, "Generalized robust Bayesian committee machine for large-scale Gaussian process regression," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3131–3140.

[10] Y. Cao and D. J. Fleet, "Generalized product of experts for automatic and principled fusion of Gaussian process predictions," 2014, *arXiv:1410.7827 [cs.LG]*.

[11] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Model learning with local Gaussian process regression," *Adv. Robot.*, vol. 23, no. 15, pp. 2015–2034, 2009.

[12] S. E. Yuksel, J. N. Wilson, and P. D. Gader, "Twenty years of mixture of experts," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1177–1193, Aug. 2012.

[13] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *IEEE Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, Aug. 2002.

[14] B. Wilcox and M. C. Yip, "SOLAR-GP: Sparse online locally adaptive regression using Gaussian processes for Bayesian robot model learning and control," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2832–2839, Apr. 2020.

[15] M. E. Kepler, A. Koppel, A. S. Bedi, and D. J. Stilwell, "Wasserstein-splitting Gaussian process regression for heterogeneous online Bayesian inference," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 9833–9840.

[16] A. K. Tanwani and S. Calinon, "Small-variance asymptotics for non-parametric online robot learning," *Int. J. Robot. Res.*, vol. 38, no. 1, pp. 3–22, 2019.

[17] T. Broderick, B. Kulis, and M. Jordan, "MAD-bayes: MAP-based asymptotic derivations from bayes," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 226–234.

[18] D. Blackwell and J. B. MacQueen, "Ferguson distributions via Pólya urn schemes," *Ann. Statist.*, vol. 1, no. 2, pp. 353–355, 1973.

[19] Y. Wang and J. Zhu, "DP-space: Bayesian nonparametric subspace clustering with small-variance asymptotics," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 862–870.

[20] L. Csató and M. Opper, "Sparse on-line Gaussian processes," *Neural Comput.*, vol. 14, no. 3, pp. 641–668, 2002.

[21] C.-A. Cheng and B. Boots, "Incremental variational sparse Gaussian process regression," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4410–4418.

[22] M. Deisenroth and J. W. Ng, "Distributed Gaussian processes," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1481–1490.

[23] D. Burt, C. E. Rasmussen, and M. van der Wilk, "Rates of convergence for sparse variational Gaussian process regression," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 862–871.

[24] S. Vijayakumar and S. Schaal, "Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space," in *Proc. 17th Int. Conf. Mach. Learn.*, 2000, pp. 288–293.

[25] P. Sears and P. Dupont, "A steerable needle technology using curved concentric tubes," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2006, pp. 2850–2856.

[26] R. J. Webster, A. M. Okamura, and N. J. Cowan, "Toward active cannulas: Miniature snake-like surgical robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2006, pp. 2857–2863.

[27] S. S. Antman, *The Special Cosserat Theory of Rods*. New York, NY, USA:Springer, 1995, pp. 259–324.

[28] D. C. Rucker, B. A. Jones, and R. J. Webster III, "A geometrically exact model for externally loaded concentric-tube continuum robots," *IEEE Trans. Robot.*, vol. 26, no. 5, pp. 769–780, Oct. 2010.

[29] C. Girerd and T. K. Morimoto, "Design and control of a hand-held concentric tube robot for minimally invasive surgery," *IEEE Trans. Robot.*, vol. 37, no. 4, pp. 1022–1038, Aug. 2021.

[30] Stanford Artificial Intelligence Laboratory *et al.*, "Robotic operating system," 2008. [Online]. Available: https://www.ros.org

[31] M. Khadem, J. O'Neill, Z. Mitros, L. da Cruz, and C. Bergeles, "Autonomous steering of concentric tube robots via nonlinear model predictive control," *IEEE Trans. Robot.*, vol. 36, no. 5, pp. 1595–1602, Oct. 2020.

[32] J. Ha, G. Fagogenis, and P. E. Dupont, "Modeling tube clearance and bounding the effect of friction in concentric tube robot kinematics," *IEEE Trans. Robot.*, vol. 35, no. 2, pp. 353–370, Apr. 2019.

[33] J. Burgner *et al.*, "A bimanual teleoperated system for endonasal skull base surgery," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2011, pp. 2517–2523.

[34] A. Capone, A. Lederer, and S. Hirche, "Gaussian process uniform error bounds with unknown hyperparameters for safety-critical applications," 2021, *arXiv:2109.02606 [cs.LG]*.