2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Sample-Efficient Robot Motion Learning using Gaussian Process Latent Variable Models

Juan Antonio Delgado-Guerrero[1], Adrià Colomé[1], and Carme Torras[1]

Abstract— **Robotic manipulators are reaching a state where we could see them in household environments in the following decade. Nevertheless, such robots need to be easy to instruct by lay people. This is why kinesthetic teaching has become very popular in recent years, in which the robot is taught a motion that is encoded as a parametric function - usually a Movement Primitive (MP)-. This approach produces trajectories that are usually suboptimal, and the robot needs to be able to improve them through trial-and-error. Such optimization is often done with Policy Search (PS) reinforcement learning, using a given reward function. PS algorithms can be classified as model-free, where neither the environment nor the reward function are modelled, or model-based, which can use a surrogate model of the reward function and/or a model for the dynamics of the task.**

However, MPs can become very high-dimensional in terms of parameters, which constitute the search space, so their optimization often requires too many samples. In this paper, we assume we have a robot motion task characterized with an MP of which we cannot model the dynamics. We build a surrogate model for the reward function, that maps an MP parameter latent space (obtained through a Mutual-information-weighted Gaussian Process Latent Variable Model) into a reward. While we do not model the task dynamics, using mutual information to shrink the task space makes it more consistent with the reward and so the policy improvement is faster in terms of sample efficiency.

I. Introduction

Learning through demonstration using kinesthetic teaching (see Fig. 1) and then improving over executions through reinforcement learning has proved to be a successful approach in several situations [1]. In Policy Search (PS), the initial demonstration is fitted into a parametric policy, which is a generative model. This model - a Movement Primitive (MP) - is used to generate samples that are evaluated with a given reward function capable of telling how good a robotic execution was. These samples and rewards are then used to obtain a new policy, which will usually maximize the expected value of the reward function given the policy. However, these approaches often require a high number of samples in order to learn sufficiently good motions in high-dimensional parameter spaces. Therefore, model-based PS is often used [2], where either a surrogate function estimating reward values or a dynamics model of the environment/robot behaviour are built. Examples of model-based PS that use both models for the dynamics and the reward functions are PILCO [3] and Black-DROPS [4].

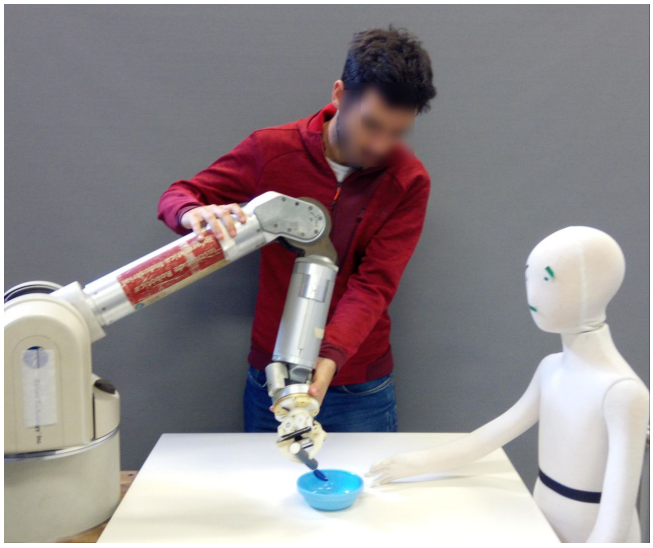

Fig. 1: Kinesthetic teaching of a feeding task.

In this paper, we assume that we are provided with an MP of a task which can be executed and evaluated, but whose dynamics can't be modelled. In such situations, building a surrogate model with Gaussian Processes for the reward function and using Upper Confidence Bound (UCB) optimization can help to converge faster to an improved solution. However, Bayesian Optimization (BO) techniques such as UCB do not scale easily to a high dimension, thus we need to perform dimensionality reduction in the MP parameter space so as to be able to carry out the optimization.

Dimensionality Reduction (DR) techniques have been used for learning robot motion [5], [6], [7], [8] and human movements [9], [10], [11], [12]. These techniques led to significant improvements by learning tasks in latent spaces. However, the smallest dimensionality that can be used is strongly limited by the DR technique used, often a linear one. This is why, in this work, we resort to Gaussian Process Latent Variable Models (GPLVM) [13] to perform such DR.

GPLVM has the advantage that it can fine-adjust the latent space dimension much more than other methods, without losing too much information. This is because of the way GPLVM is built: instead of looking for a projection from the original space to the latent space, it generalizes Dual Probabilistic Principal Component Analysis (DPPCA) and

This work was partially developed in the context of the project CLOTHILDE ("CLOTH manIpulation Learning from DEmonstrations"), which has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Advanced Grant agreement No 741930). This work is also supported by the Spanish State Research Agency through the María de Maeztu Seal of Excellence to IRI MDM-2016-0656.

[1]Institut de Robòtica i Informàtica Industrial (IRI), CSIC-UPC, Barcelona, Spain. [jdelgado,acolome,torras]@iri.upc.edu.

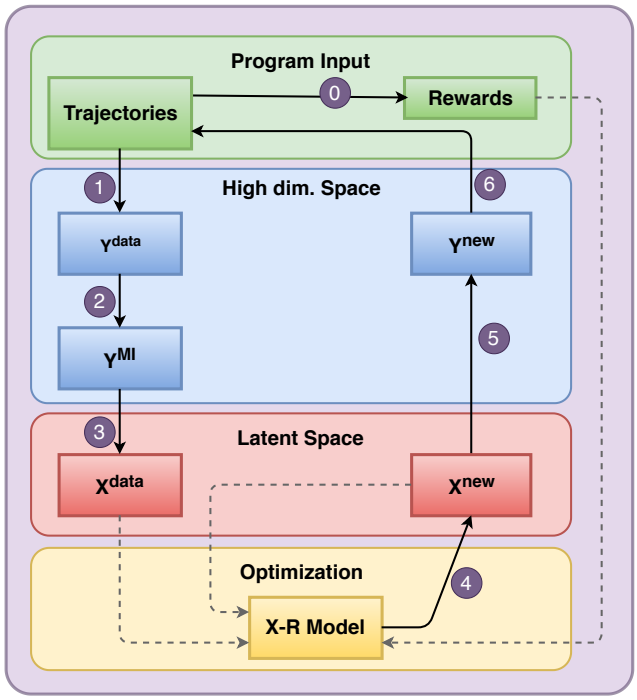

Fig. 2: Global scheme of the proposed approach. Trajectories are evaluated (0) and fit (1) into data vectors $\mathbf{Y}^{\text{data}}$ as ProMP parameters, these data are weighted (2) dimension-wise with their mutual information with rewards. GPLVM (3) finds a latent data model with variables $\mathbf{X}^{\text{data}}$. These data, together with their rewards, are used to build a surrogate model of the reward function in the latent space, which can estimate the reward directly from the latent space. UCB exploration is used (4) to generate new samples in the latent space, which are then used to predict (5) their respective high-dimensional state value and then executed (6) as trajectories and evaluated. The outputs of these evaluations and their generators in the latent space are sent back to the surrogate model of the reward, which will be updated.

finds the values of the latent space variables that maximize the likelihood with respect to the data. This, together with the sampling efficiency of Gaussian Processes (GP), allows us to reduce the dimensionality of the search space by at least one order of magnitude, significantly speeding up the convergence to a better motion policy. Additionally, we use the Mutual Information (MI) [14] between the data dimensions and the reward in order to weigh data and prioritize the fitting of the reward function instead of the MP parameters. Finally, we can use UCB to decide which samples to evaluate from the latent space, project them upwards, execute, evaluate and then update a surrogate model of the reward function that maps the latent space parameters to the original reward function, considered as a black box. Fig. 2 shows a schematic view of the proposed method.

This paper is organized as follows: Section II briefly introduces the concepts used in the paper, such as Probabilistic Movement Primitives (ProMPs) [15], Gaussian Processes (GP) [16], Gaussian Process Latent Variable Models (GPLVM) [13], Mutual Information (MI) [14] and Upper Confidence Bound (UCB) [17], [18]. Section III defines the proposed approach and Section IV presents the results obtained with this method. Section V concludes the paper and proposes future directions.

II. Preliminaries

A. Probabilistic Movement Primitives

ProMPs [15] are an overall approach to model, encode and learn a set of similar motion trajectories that present time-dependent variances over time. Given a number of basis functions per Degrees of Freedom (DoF), N_f, ProMPs use time-dependent Gaussian kernels $\mathbf{\Phi}_t$ to encode the state of a trajectory, $\mathbf{\Phi}_t$ being the vector of normalized kernel basis functions (e.g., uniformly distributed Gaussian basis function over time). Thus, the position and/or velocity state vector $\mathbf{z}_t$ can be represented as

$$\mathbf{z}_t = \mathbf{\Psi}_t^T \boldsymbol{\omega} + \boldsymbol{\epsilon}_z, \tag{1}$$

where $\mathbf{\Psi}_t^T = I_r \otimes \mathbf{\Phi}_t^T$, I_r being the r-dimensional identity matrix, with r the number of DoFs of the robot, and $\mathbf{\Phi}_t$ an N_f-dimensional column vector with the Gaussian kernels associated to one DoF at time t. Furthermore, $\boldsymbol{\epsilon}_z$ is a zero-mean Gaussian noise and the weights $\boldsymbol{\omega}$ are also treated as random variables with a distribution $p(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}|\boldsymbol{\mu}_\omega, \mathbf{\Sigma}_\omega)$. Given a set of demonstration trajectories $\boldsymbol{\tau}_n = \{\mathbf{z}_t^n\}_{t=1..N_t}$, $n = 1..N$, this distribution can be fitted by obtaining the weights $\boldsymbol{\omega}_n$ of each demonstration through least squares. Afterwards, the parameters of the distribution $\boldsymbol{\theta} = \{\boldsymbol{\mu}_\omega, \mathbf{\Sigma}_\omega, \mathbf{\Sigma}_z\}$, $\mathbf{\Sigma}_z$ being the state noise covariance, are fitted by means of a maximum likelihood estimate, i.e., computing the mean and covariance of the samples $\{\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_N\}$. Therefore, the probability of observing a $\mathbf{z}_t$ is:

$$p(\mathbf{z}_t; \boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{z}_t | \mathbf{\Phi}_t^T \boldsymbol{\mu}_\omega, \mathbf{\Sigma}_z + \mathbf{\Phi}_t^T \mathbf{\Sigma}_\omega \mathbf{\Phi}_t\right) \tag{2}$$

Due to the probabilistic representation, ProMPs can represent motion variability while keeping other MP properties such as rescalation and linear representation with respect to parameters.

B. Gaussian Processes

A Gaussian Process (GP) [16] f is an infinite-dimension stochastic process such that, for any finite set of indices $x_1, ..., x_n$, the random variables $f(x_1), ..., f(x_n)$ have joint Gaussian distributions. A GP is completely defined by its mean function m and covariance function k, which is symmetric and positive semi-definite:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{3}$$

This is the generalization of the multivariate Gaussian distribution, over vectors, to an infinite-dimension stochastic process, over functions.

For convenience, the mean function is usually assumed to be the zero function, $m(\mathbf{x}) = 0$. On the other hand, many possibilities can be found in the literature for defining the covariance function k. In this work, the squared exponential kernel, combined with a vector of automatic relevance determination, has been used for this purpose:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \sigma^2 \exp\left(-\frac{1}{2}(\mathbf{x_i} - \mathbf{x_j})^T \text{diag}(\boldsymbol{\ell})^{-2} (\mathbf{x_i} - \mathbf{x_j})\right), \tag{4}$$

where σ is the kernel variance parameter and $\boldsymbol{\ell}$ is the length-scale vector parameter.

Furthermore, Gaussian processes are useful for regression models, $\mathbf{y} = f(\mathbf{x}) + \epsilon$. Thus, considering a set of N observations in matrix form: $\{\mathbf{X}, \mathbf{Y}\}$, with $\mathbf{X} \in \mathbb{R}^{N \times Q}$, $\mathbf{Y} \in \mathbb{R}^{N \times D}$, f can be used to predict the value of $\mathbf{y}_{N+1}$, given $\mathbf{x}_{N+1}$, and a noise Gaussian distribution ϵ. By leveraging the properties of f and the Gaussian identities, one can arrive to the expression:

$$\begin{aligned} &P(\mathbf{y}_{N+1}|\mathbf{X}, \mathbf{Y}, \mathbf{x}_{N+1}) = \\ &= \mathcal{N}(\mu_t(\mathbf{x}_{N+1}), \sigma_t^2(\mathbf{x}_{N+1}) + \sigma_{noise}^2), \end{aligned} \tag{5}$$

where

$$\mu_t(\mathbf{x}_{N+1}) = \mathbf{k}^T[\mathbf{K} + \sigma_{noise}^2 I_N]^{-1}\mathbf{Y} \tag{6}$$

$$\sigma_t^2(\mathbf{x}_{N+1}) = k(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) - \mathbf{k}^T[\mathbf{K} + \sigma_{noise}^2 I_N]^{-1}\mathbf{k} \tag{7}$$

$$\mathbf{K}_{i,j} = k(x_i, x_j) \quad i, j = 1..N \tag{8}$$

$$\mathbf{k}_i = k(x_{N+1}, x_i) \quad i = 1..N \tag{9}$$

C. Gaussian Process Latent Variable Models

GPLVMs can be thought as the combination of GPs, explained in section II-B, and latent variable models. These models, as a type of feature extraction approach, relate through a set of parameters the observed data, $\mathbf{Y} \in \mathbb{R}^{N \times D}$, with a set of so-called latent or hidden variables, $\mathbf{X} \in \mathbb{R}^{N \times Q}$, being $Q \ll D$ for the purpose of dimensionality reduction. In this way, GPLVMs define a generative mapping from the latent space to the observation space, whose variable responses are said to be governed by the latent ones.

Besides, GPLVMs are formulated as a non-linear generalization of Probabilistic Principal Component Analysis (PPCA) [13]. Specifically, GPLVMs arise directly from the formulation of Dual PPCA models, by replacing the inner product kernel with a non-linear covariance function. These approaches marginalize the parameters and optimize the latent variables. Therefore, the marginal likelihood function $p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\theta})$ can be expressed as:

$$p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\theta}) = \prod_{d=1}^{D} p(\mathbf{y}_{:,d}|\mathbf{X}), \tag{10}$$

where $\mathbf{y}_{:,d}$ is the $d-$th column of the data matrix $\mathbf{Y}$, corresponding to the $d-$th dimension, and $\mathbf{y}_{:,d}|\mathbf{X} \sim \mathcal{N}(\mathbf{y}_{:,d}|\mathbf{0}, \mathbf{K} + \sigma_{noise}^2 I)$

Finally, the methodology for training the GPLVM is to find the maximum a posteriori estimate of $\mathbf{X}$, maximizing Eq. (10) with respect to the latent variable values and noise parameters.

GPLVM results in a projection from the higher dimensional space to the latent space, without providing a projection mapping by itself. Not restricting the mapping to a certain expression allows GPLVM to fine-tune the values of the latent space further. Besides, GPLVM does provide a tool for estimating the higher-dimensional variable $\mathbf{y}$ with respect to the lower dimensional $\mathbf{x}$, by also using Eq. (5).

D. Mutual Information

Mutual Information (MI) [14] is a widespread non-negative symmetric statistic for quantifying the degree of dependence between two random variables, being zero if and only if such variables are independent. In other words, it measures the amount of information shared by the variables, reflecting how much information about one of them may arise, from the knowledge of the other one.

Formally, this concept, which is grounded in information theory, is defined as the relative entropy between the joint probability, and the product distributions. Therefore, the mutual information $I(X;Y)$ between two continuous random variables with joint density $p(X,Y)$ is:

$$I(X;Y) = \int\int p(x,y) \log(\tfrac{p(x,y)}{p(x)p(y)}) dxdy \tag{11}$$

where $p(x)$ and $p(y)$ are the marginal probabilities with respect to X and Y, respectively.

Normally, and so it happens in our case, the joint probability distribution is not known, being only available some sampled data of the form $\{x_i, y_i\}_{i \in [N]}$. In such cases, MI must be estimated from this data set. One straightforward approach for this purpose consists in partitioning the variables domain into bins of finite size, and approximate Eq. (11), as in [19].

E. Bayesian Optimization and Upper Confidence Bound

Bayesian Optimization [18] addresses the issue of finding the extrema of objective functions, which have no closed-form expression, unknown derivatives and convexity, or are costly to evaluate, as in our case. In such cases, these approaches result efficient in terms of the number of function evaluations required to reach convergence [20].

These methods mainly consist of two components: a stochastic surrogate model fitting the target function f, and an acquisition function. On the one hand, the surrogate model takes advantage of the information of accumulated observations to generate a posterior distribution from a prior distribution, by means e.g. of Gaussian process, as in section II-B. The surrogate function will usually have more uncertainty (variance) on those unexplored areas or where its value is much non-deterministic.

On the other hand, the acquisition function, defined in a search space $\boldsymbol{\Omega_X} \subset \mathbb{R}^{N \times Q}$, uses the surrogate model to define the utility of evaluating each point of $\boldsymbol{\Omega_X}$, giving more importance to points which are likely to have high objective function values, considering both the surrogate model prediction and its uncertainty.

Thus, the result of the maximization of the acquisition function is selected as the next point of the objective function to be evaluated, being the surrogate model updated accordingly right afterwards. In this way, the acquisition function is responsible to lead the search for the optimum of the objective function, in a trade-off frame between exploration and exploitation.

Upper Confidence Bound is a very intuitive [2] and efficient [21] acquisition function method, defined by:

$$\text{UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\sigma(\mathbf{x}), \tag{12}$$

where κ is a parameter (left to the user) which controls the importance of exploitation. The new samples are then generated as:

$$\mathbf{x}_{\text{sample}} = \text{argmax}_{\mathbf{x}\in\mathbf{\Omega}_X} \text{UCB}(\mathbf{x}) \tag{13}$$

This method results in choosing to sample the point which presents the highest mean plus κ standard deviations value on the surrogate function model.

III. Learning in Latent Spaces through GPLVM

In this section, using the concepts defined in Sec. II we present an approach that is capable of learning high-dimensional robot motion policies with very few samples.

Given a set of trajectories $\boldsymbol{\tau}_n = \{\mathbf{z}_j^n\}$, for trajectories $n = 1..N$ and timesteps $j = 1..N_t$, we will firstly fit those trajectories to MPs parameters by obtaining, for each trajectory, $\boldsymbol{\omega}_n$ that adjust the trajectory according to (1) (by using least-squares). These weight vectors, capable of fitting each individual trajectory given the previously fixed ProMP kernels, are then gathered together as the rows of our data matrix $\mathbf{Y}$. Therefore, we are performing Dimensionality Reduction (DR) in the parameter space of the MP, rather than directly in the space of degrees of freedom of the robot, as other approaches do [6], [8]. While performing DR in the space of degrees of freedom is advantageous in that it provides qualitative information that is directly interpretable, performing DR in the MP's parameter space permits to reduce further and fine-tune the dimensionality of the latent space [22], which is one of the goals of this work.

We will then use GPLVM on the data stored in matrix $\mathbf{Y}$. However, while GPLVM was initially conceived for data visualization [23], we are aiming at further improving the model given the evaluations of each trajectory through a reward function:

$$\begin{aligned} R : \mathbb{R}^D &\longrightarrow \mathbb{R} \\ \mathbf{y} &\longmapsto R(\mathbf{y}). \end{aligned} \tag{14}$$

Therefore, when deriving the GPLVM, we will weigh the data by using $\mathbf{Y}^{\text{MI}} = \mathbf{Y} \cdot \mathbf{MI}^{1/2}$ instead of data $\mathbf{Y}$. Here, $\mathbf{MI}$ is the mutual information between the reward function and each column of the data - i.e., each parameter of the parameter vector $\boldsymbol{\omega}$ of the MP. The $\mathbf{MI}$ will then be higher when there is a relation between such parameter (column) and the reward, and smaller when such parameter does not have much influence on the reward. Indeed, if we use this weighting $M_d = \text{MI}(\mathbf{y}_{:,d}, \mathbf{R})$ on the log-likelihood, we can see that optimizing the weighted log-likelihood is equivalent to using such weighted data as input:

$$\log\, p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\theta}) = \log \prod_{d=1}^{D} p(\mathbf{y}_{:,d}|\mathbf{X})^{M_d}, \tag{15}$$

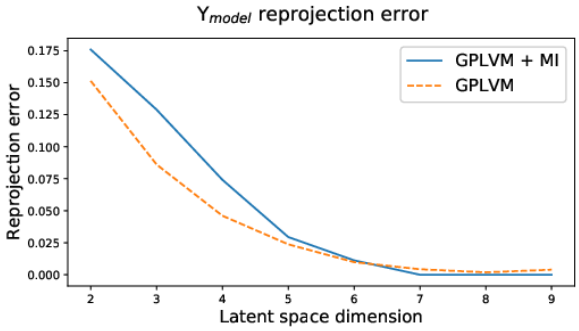

Fig. 3: Mean Euclidean norm of the error on the reprojection of each data trajectory within a training dataset, comparing GPLVM with MI-weighted GPLVM.

which can be developed into:

$$\begin{aligned} &\log \prod_{d=1}^{D} p(\mathbf{y}_{:,d}|\mathbf{X})^{M_d} = \\ &\sum_{d=1}^{D} M_d \left[-\frac{N}{2}\log 2\pi - \log|\mathbf{K}| - \frac{1}{2}\mathbf{y}_{:,d}^T \mathbf{K}^{-1} \mathbf{y}_{:,d}^T \right] = \\ &= (\tfrac{N}{2}\log\, 2\pi - \log\, |\mathbf{K}|) \sum_{d=1}^{D} M_d \\ &\quad - \sum_{d=1}^{D} \left(\mathbf{y}_{:,d}\sqrt{M_d}\right)^T \mathbf{K}^{-1} \left(\mathbf{y}_{:,d}\sqrt{M_d}\right) = \\ &= \sum_{d=1}^{D} M_d \cdot C - \sum_{d=1}^{D} \text{tr}\left(\mathbf{Y}^{\text{MI}T} \mathbf{K}^{-1} \mathbf{Y}^{\text{MI}T}\right) \end{aligned} \tag{16}$$

where $C = \frac{N}{2}\log\, 2\pi - \log\, |\mathbf{K}|$ is a constant, $\mathbf{Y}^{\text{MI}} = \mathbf{Y} \cdot \text{diag}(M_{1..D})$, and $\text{diag}(M_{1..D})$ is a diagonal matrix with the values of the mutual information between column d of $\mathbf{Y}$ and the rewards at each d-th diagonal position, for $d = 1..D$.

Therefore, using such weighting on the input data for the GPLVM allows us to have a significantly smaller error when calculating the reward of the mean predicted reprojection for each training motion $\mathbf{y}_n$, while not losing much precision on such reprojection. In Fig. 3, we see the reprojection error from a dataset using GPLVM with and without MI weighting for the training dataset, while 30% of data was left for validation purposes as shown in Fig. 4. There, we can observe that, while the reprojected training points present a slightly better fitting with MI, this result is worse for the validation datapoints. However, if we pay attention to the reward evaluation of the latent space datapoints, reprojected again to the higher-dimensional space, this gives an idea of how the rewards are affected by the coding/decoding operation. We see in Fig. 5 - in log10 scale - that there is a slight improvement on the training dataset by using MI, and at least one order of magnitude of improvement on the reward evaluations for the validation dataset, as observed in Fig. 6. This means that using such MI to weigh the GPLVM input results in a model that more reliably represents the reward function, for which we will build a surrogate model.

Once having built a GPLVM to obtain a proper latent space representation of data, we will build a surrogate model that

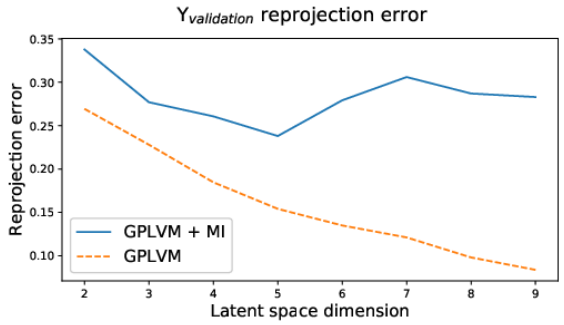

Fig. 4: Mean Euclidean norm of the error on the reprojection of each data trajectory within a validation dataset, comparing GPLVM with MI-weighted GPLVM.

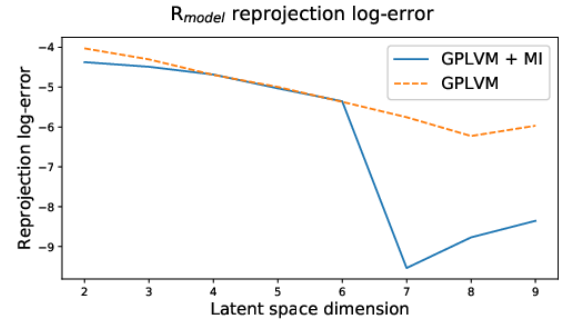

Fig. 5: Mean Euclidean norm of the error (in log10 scale) on the evaluation of the reward function of the reprojected value of each data trajectory within a training dataset, comparing GPLVM with MI-weighted GPLVM.

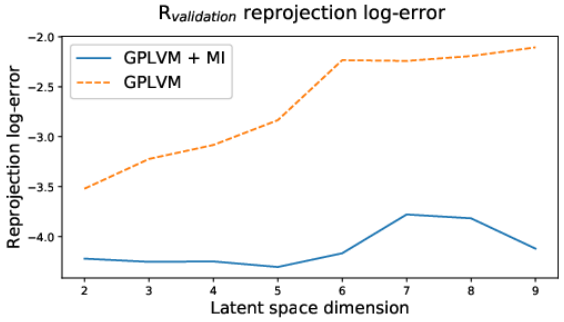

Fig. 6: Mean Euclidean norm of the error (in log10 scale) on the evaluation of the reward function of the reprojected value of each data trajectory within a validation dataset, comparing GPLVM with MI-weighted GPLVM.

estimates the reward function to be evaluated for the latent-space variables rather than in the original higher-dimensional space:

$$\begin{aligned} \hat{R} : \mathbb{R}^Q &\longrightarrow \mathbb{R} \\ \mathbf{x} &\longmapsto \hat{R}(\mathbf{x}). \end{aligned} \tag{17}$$

In order to build the surrogate model for the latent space reward function, we will rely on a Gaussian Process (see Sec. II-B) that uses the latent space variables and their evaluations.

For learning and improving the trajectories, we will then use UCB (see Sec. II-E) to generate new samples. The UCB will be using the mean and variance provided by the GP

Algorithm 1

Input:

Trajectory data τ_{jl}^n , $n = 1..N$, $j = 1..N_t$, $l = 1..N_{\text{DOF}}$
Demonstrated trajectories rewards R_n, $n = 1..N$
GPLVM latent space dimension Q
ProMPs' kernel matrix $\mathbf{\Psi}$

Compute weights $\boldsymbol{\omega}_n$ with Eq.(1)
Assign $\mathbf{Y}_n \leftarrow \boldsymbol{\omega}_n$
Compute MI: $M_d = \text{MI}(\mathbf{y}_{:,d}, \mathbf{R})$, $d = 1..D$
Reassign GPLVM input data $\mathbf{Y}^{\text{MI}} \leftarrow \mathbf{Y} \cdot \text{diag}(M_{1..D})$
Perform GPLVM($\mathbf{Y}_n^{\text{MI}}$) and obtain $\mathbf{X}_n$
Train $\mathbf{X}$-R Regression Model $\hat{R} \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot))$
Define Search region $\mathbf{\Omega_X} \subset \mathbb{R}^{N \times Q}$
for $k = 1..N_{new}$ **do**
    Define $\text{UCB}(\mathbf{x}) = \mu_{k-1}(\mathbf{x}) + \kappa\sigma_{k-1}(\mathbf{x})$
    Generate new sample $\mathbf{x}_k = \arg\max_{\mathbf{x} \in \mathbf{\Omega}_X} \text{UCB}(x)$
    Project $\mathbf{x}_k$ to $\tilde{\mathbf{y}}_k$ with Eq. (5)
    Execute $\tilde{\mathbf{y}}_k$ and evaluate $R(\tilde{\mathbf{y}}_k(\mathbf{x}_k))$
    Update f, μ_k, and σ_k with $\mathbf{x}_k$ and $R(\mathbf{x}_k)$
end for

surrogate model. As already mentioned in Sec. II-E, UCB suggests to evaluate points, according to the maximization of Eq. (12), in a certain search space $\mathbf{\Omega_X}$.

Each sample $\mathbf{x}_{\text{sample}}$ in the latent space will be used to estimate the corresponding value in the higher-dimensional space $\hat{\mathbf{y}}(\mathbf{x})$ with Eq. (5), which will be executed and evaluated, giving us the real value of the reward function R associated with $\mathbf{x}_{\text{sample}}$. This new sample, and the associated reward, will then be added to the surrogate model, that will be updated, before generating new samples. The process is repeated until convergence or a certain number of samples have been executed. Algorithm 1 displays the procedure of the proposed method, while Fig. 2 shows a more schematic view.

In this work, the search space $\mathbf{\Omega_X}$ has been defined as the minimum axis-aligned hyperrectangle that contains all latent data, as suggested in [18]. In order to select candidates, the exploration parameter κ in Eq. (12) has been fixed for simplification purposes ($\kappa = 1$). However, less naive methods for selecting this parameter can be found, as in [17], [24].

IV. Experimentation

In order to show the performance of the proposed method, we taught a Barret WAM robot 50 trajectories of feeding a mannequin (see Fig. 1) with the robot going from one initial area (start position) to a final position area (mannequin head position), getting food from random positions on the table. The trajectories of the robot's end-effector can be seen in Fig. 7. As input data, we used the position $\{x, y, z\}$ of the robot's end-effector to represent the trajectory, and 11 Gaussians per Cartesian dimension, resulting in a 33-dimensional parameter space. Then, we placed a bowl at a particular position and we defined a reward function by calculating the Euclidean

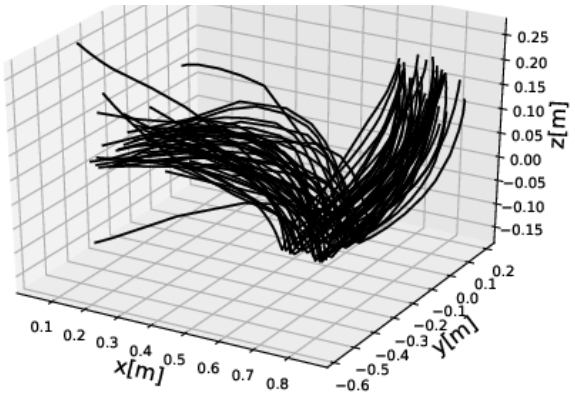

Fig. 7: Robot trajectories obtained through kinesthetic teaching in the 3D space. Trajectories start from the left side.

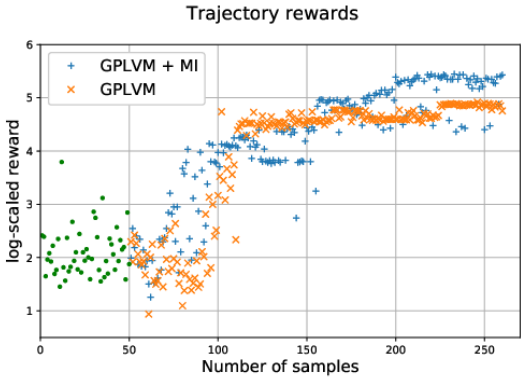

Fig. 8: Samples and their real rewards (in log10 scale) with GPLVM and UCB learning, by using MI weighting and not. The initial 50 samples shown in green correspond to the training dataset. Note that the first 10 samples are re-sampled demonstrations in order to better fit the surrogate function.

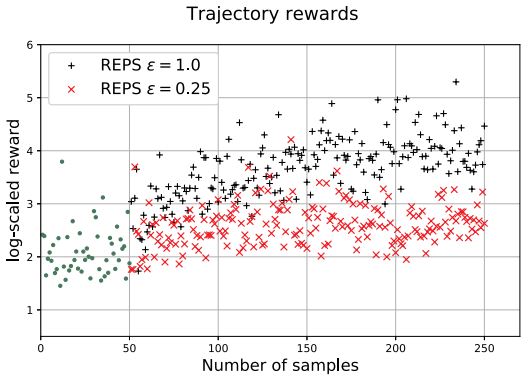

Fig. 9: Samples and their real rewards (in log10 scale) with REPS with two different KL divergence bounds, $\epsilon_{KL} = 0.25$ and $\epsilon_{KL} = 1.0$. The initial 50 samples shown in green correspond to the training dataset.

distance between the lowest-height point of each trajectory, the contact point $\mathbf{c_p}$, and the bowl center, which is the objective point $\mathbf{o_p}$:

$$R = -\text{dist}(\mathbf{c_p}, \mathbf{o_p})^2 \tag{18}$$

In order to use our proposed method in Algorithm 1, we initially found the problem that the $\mathbf{X}$-R regression model was identifying the latent data with both originally demonstrated motions' rewards and reprojected samples' rewards, with poor results. Therefore, as reprojected trajectories were similar, we then performed and evaluated the reprojection of 10 out of the 50 original demonstrations, by reprojecting their associated latent space point to the higher-dimensional space and evaluating it. Then, we built the reward's surrogate model with those 10 resampled motions, and iteratively updated it, with new samples and evaluations through UCB. The results, both with the MI weighting on the data for GPLVM and without it, are shown in Fig. 8, where we see there is an improvement when using MI to correct data.

Moreover, we compared these two methods against a state-of-the-art policy search algorithm: Relative Entropy Policy Search (REPS) [25]. REPS's optimization maximizes the expected reward of a stochastic policy, subject to obtaining a stochastic policy and keeping the Kullback-Leibler divergence between the old and new policy bounded by a certain parameter ϵ_{KL}. Same scaled Fig. 9, shows that, while REPS optimizes the trajectory and reaches a similar long-term reward, we see that it requires more samples than our proposed method for $\epsilon_{KL} = 0.25, 1.0$. We could check that trajectories with a high reward were desirable. In fact, $\log_{10}(-R) > 4$ implied millimetric precision.

V. Conclusion

In this paper, we have proposed an approach to use Bayesian optimization-based policy search to optimize robot motion trajectories with a surrogate model of the reward function. To this end, we have used GPLVM to reduce the parameter space of robot movement primitives to a much lower-dimensional space, e.g., going down from a 33-dimensional parameter space to a 4-dimensional space. In such latent space, optimizing the robot's policy is much more sample efficient, resulting in a much faster convergence to an optimal solution, also thanks to weighting the GPLVM with the mutual information of each dimension with respect to the rewards observed. The experimental results provided show that this optimization method converges much faster than other state-of-the-art methods, such as REPS.

While the initial latent space may restrict the search space for the policy learning, the prediction used to obtain the higher-dimensional variables from the latent space samples can be perturbed, so that exploration would also take place outside the initial Q-dimensional latent space, and the GPLVM could be updated after a certain number of samples. This is left for future work, as well as other approaches to adapt UCB to higher-dimensional spaces without sampling too much on the extremes of the sampling regions. Moreover, we plan to expand this work to make it adaptable to environmental changes that the robot might perceive, using GPLVM extensions such as Bayesian GPLVM [26],[27], and to improve the discussion regarding the advantages of our approach with respect to other state-of-the-art methods.

REFERENCES

[1] M. P. Deisenroth, G. Neumann and J. Peters, "A survey on Policy Search for Robotics". *Foundations and Trends in Robotics*, vol 2, pp 1-142, 2013.

[2] K. Chatzilygeroudis, V. Vassiliades, F. Stulp, S. Calinon, and J. B. Mouret, "A survey on policy search algorithms for learning robot controllers in a handful of trials". arXiv preprint arXiv:1807.02303, 2018.

[3] M. Deisenroth and C. E. Rasmussen. "PILCO: A model-based and data-efficient approach to policy search". *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465-472, 2011.

[4] K. Chatzilygeroudis, R. Rama, R. Kaushik, D. Goepp, V. Vassiliades and J. B. Mouret, "Black-box data-efficient policy search for robotics". *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 51-58, 2017.

[5] S. Bitzer and S. Vijayakumar, "Latent spaces for dynamic movement primitives." *IEEE-RAS Int. Conf. on Humanoid Robots*, pp. 574 - 581, 2009.

[6] A. Colomé and C. Torras. "Dimensionality reduction for dynamic movement primitives and application to bimanual manipulation of clothes". *IEEE Transactions on Robotics*, vol. 34, no .3, pp. 602-615, 2018.

[7] A. Colomé, G. Neumann, J. Peters and C. Torras. "Dimensionality reduction for probabilistic movement primitives", *IEEE-RAS Humanoid Robots*, pp. 794-800, 2014.

[8] K. S. Luck, G. Neumann, E. Berger, J. Peters, and H. Ben Amor, "Latent space policy search for robotics." *IEEE/RSJ Int. Conf. on Intelligent Robots (IROS)*, pp. 1434-1440, 2014.

[9] A. Sadamani, A. Ghodsi, D. Kulic, "Discriminative funcitonal analysis of human movements." *Pattern Recognition Letters*, vol. 34, pp. 1829-1839, 2013.

[10] G. Averta, C. Della Santina, E. Battaglia, F. Felici, M, Bianchi, and A. Bicchi, "Unvealing the principal modes of human upper limb movements through functional analysis". *Frontiers in Robotics and AI*, 4:37, 2017.

[11] N. Coffey, A. Harrison, O. Donoghue and K. Hayes, "Common functional principal components analysis: A new approach to analyzing human movement data". *Human movement science*, vol. 30, pp. 1144-1166, 2011.

[12] W. Dai, "FPCA Based Human-like Trajectory Generating". 2013

[13] N. Lawrence, "Probabilistic non-linear principal component analysis with Gaussian process latent variable models", *Journal of machine learning research*, vol. 6, no. Nov, pp. 1783–1816, 2005.

[14] L. F. Kozachenko, N. N. Leonenko, "Sample Estimate of the Entropy of a Random Vector", Probl. Peredachi Inf., vol. 23, no. 2 pp. 9-16, 1987.

[15] A. Paraschos, G Neumann, C. Daniel, and J. Peters, "Probabilistic movement primitives". *In Advances in NIPS*, pp. 2616-2624, 2013.

[16] C. E. Rasmussen and C. K. I. Williams, "Gaussian Processes for Machine Learning", *the MIT Press*, 2006. ISBN 026218253X.

[17] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: no regret and experimental design" *International Conference on Machine Learning (ICML)*, pp 1015-1022, 2010.

[18] E. Brochu, V. M. Cora, N. de Freitas, "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning", arXiv preprint arXiv:1012.2599, 2010.

[19] A. Kraskov, H. Stögbauer, and P. Grassberger. "Estimating Mutual Information", *Physical Review E*, no. 69, 066138, 2004.

[20] J. Mockus. "Application of Bayesian approach to numerical methods of global and stochastic optimization", *Journal of Global Optimization*, vol. 4, no. 4, pp. 347-365, 1994.

[21] P. Hennig and C.J. Schuler. "Entropy search for information-efficient global optimization", *Journal of Machine Learning Research*, vol. 13, no. Jun, pp.1809-1837, 2012.

[22] A. Colomé and C. Torras. "Dimensionality reduction in learning Gaussian mixture models of movement primitives for contextualized action selection and adaptation", *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3922-3929, 2018.

[23] N. Lawrence, "Gaussian Process Latent Variable Models for Visualisation of High Dimensional" *International Conference on Neural Information Processing Systems*, 2004.

[24] S. Grünewälder, J.-Y. Audibert, M. Opper and J. Shawe-Taylor, "Regret Bounds for Gaussian Process Bandit Problems". *Journal of Machine Learning Research*, no 9. pp 273-280, 2010.

[25] J. Peters, K. Mülling and Y. Altün, "Relative Entropy Policy Search". *National Conf. on Artificial Intelligence*, track 15, pp. 182-189, 2011.

[26] P. Li, S. Chen, "A review on Gaussian Process Latent Variable Models". *CAAI Transactions on Intelligence Technology*, vol. 1, no. 4, pp. 366-376, 2016.

[27] M. K. Titsias, N. D. Lawrence, "Bayesian Gaussian Process Latent Variable Model". *International Conference on Artificial Intelligence and Statistics*, vol. 9 of JMLR:W&CP 9, 2010.