Dimensionality Reduction in Learning Gaussian Mixture Models of Movement Primitives for Contextualized Action Selection and Adaptation

Adrià Colomé[1] and Carme Torras[1]

***Abstract*— Robotic manipulation often requires adaptation to changing environments. Such changes can be represented by a certain number of contextual variables that may be observed or sensed in different manners. When learning and representing robot motion –usually with movement primitives–, it is desirable to adapt the learned behaviors to the current context. Moreover, different actions or motions can be considered in the same framework, using contextualization to decide which action applies to which situation. Such frameworks, however, may easily become large-dimensional, thus requiring to reduce the dimensionality of the parameters space, as well as the amount of data needed to generate and improve the model over experience.**

In this paper, we propose an approach to obtain a generative model from a set of actions that share a common feature. Such feature, namely a contextual variable, is plugged into the model to generate motion. We encode the data with a Gaussian Mixture Model in the parameter space of Probabilistic Movement Primitives (ProMPs), after performing Dimensionality Reduction (DR) on such parameter space, in a similar fashion as in [1]. We append the contextual variable to the parameter space and obtain the number of Gaussian components, i.e., different actions in a dataset, through Persistent Homology. Then, using multimodal Gaussian Mixture Regression (GMR) [2], we can retrieve the most likely actions given a contextual situation and execute them. After actions are executed, we use Reward-Weighted Responsibility GMM (RWR-GMM) update the model after each execution. Experimentation in 3 scenarios shows that the method drastically reduces the dimensionality of the parameter space, thus implementing both action selection and adaptation to a changing situation in an efficient way.

I. Introduction

Movement primitives (MP) are nowadays the standard tool for learning robotic tasks, due to their versatile approximation of robot motion that allows for different operations, as well as learning. In particular, robot motion learning is often performed by direct Policy Search (PS) [3], where the policy parameters - the MP parameters - are optimized so as to maximize the expected reward of executing the newly-generated motions. However, many robot applications need to adapt to changing situations, i.e.: contextual features. An example would be to learn to feed a person from a plate, while changing their positions (see Fig. 1) from a bottle into a glass. If we change the glass position, the motion must adapt to the new contextual situation. Therefore, a lot of effort has been put during the last years on being able to learn adaptive behaviors. In [4], a popular PS algorithm named Relative Entropy Policy Search (REPS) was extended to include contextual dependency on the newly obtained policy. In [5] this work is extended by combining such methodology with Covariance Matrix Adaptation [6] to overcome the greedy-ness of some contextual PS algorithms. In [7], a hierarchical structure of actions is learned, representing a probability distribution over the contextual variables. This structure is then updated, taking care of keeping locality below controlled margin by setting entropy and Kullback-Leibler divergence bounds between the newly generated policies and previous ones.

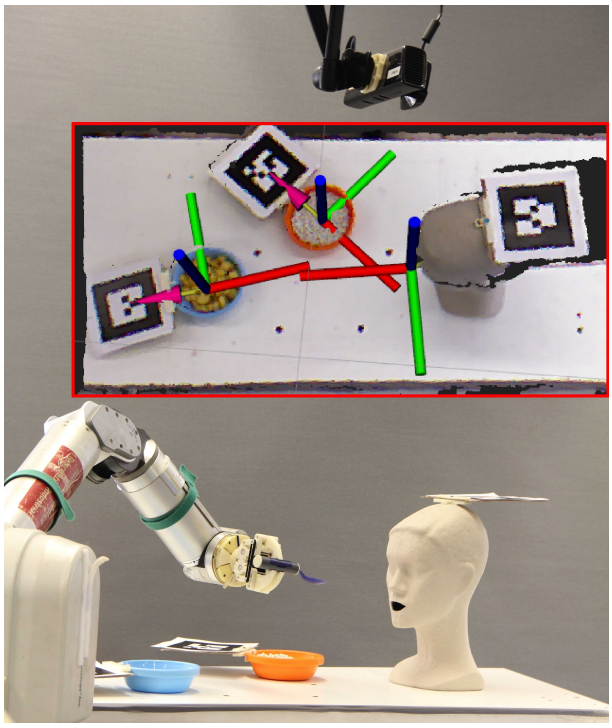

Fig. 1. Feeding task setup. The arm picks up food from the user-preferred plate and brings it to the mouth of the mannequin's head. The vision from a Kinect camera is shown in a red frame.

Manuscript received: February 24th, 2018; Revised: May 29th, 2018; Accepted: July 3rd, Year.

This paper was recommended for publication by Editor Dongheui Lee upon evaluation of the Associate Editor and Reviewers' comments.

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 and by European project I-DRESS (PCIN-2015-147).

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

Having several alternatives for similar motions that share a contextual variable - which is the ultimate responsible for deciding which action to take - can result in a very high dimensionality of such representation. Therefore, it is interesting to keep the information of a contextual, hierarchical structure of movement primitives in a smaller dimensional structure. The work in [8] presents a GMM in the time-space domain that is combined with a Hidden Semi-Markov Model (HSMM) that helps to transition between components of the GMM along time. Other contextual adaptations in literature include force sensors to adapt motion [9] or build a dynamical system that can be modulated [10].

This paper is structured as follows: In Section II, we briefly introduce Probabilistic Movement Primitives [11], to then adapt the authors' work [1] in Section II-A to perform linear DR in the parameter space of a ProMP. In Sections III-A, III-B, we introduce Gaussian Mixture Models (GMM) [12] and Regression (GMR). This paper proposes a novel approach in which the MPs parameters are represented by using a combination of latent space MP weights with the context variables. This allows to then infer a probability distribution of MPs given a context, which is adapted to the contextual observations. We also introduce Persistent Homology [13] to find the number of Gaussians (corresponding to different actions) composing the GMM. This allows the proposed method to work in those cases where no information regarding the number of actions is given a priori, as seen in the experiments. Using the responsibilities provided by the conditional GMM obtained, the most suitable action can be selected, executed and evaluated, and the entire model can be updated, following the guidelines presented in [14], as described in Section III-C, which are adapted for iterative optimization in Section III-D. An overview of the method and its experimental validation follow in Sections IV and V, respectively.

II. Probabilistic Movement Primitives

ProMPs are a general approach to learn and encode a set of similar motion trajectories that present time-dependent variances over time. Given a number of basis functions per 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 $\mathbf{y}_t$ can be represented as

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

where $\mathbf{\Psi}_t^T = I_d \otimes \mathbf{\Phi}_t^T$, I_d being the d-dimensional identity matrix and $\mathbf{\Phi}_t$ an N_f-dimensional column vector with the Gaussian kernels associated to one DoF at time t. Moreover, $\boldsymbol{\epsilon}_y \sim \mathcal{N}(0, \mathbf{\Sigma}_y)$ is a zero-mean Gaussian noise and the weights $\boldsymbol{\omega}$ are also treated as random variables with a distribution

$$p(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}|\boldsymbol{\mu}_\omega, \mathbf{\Sigma}_\omega). \tag{2}$$

This distribution can be fitted, given a set of demonstration trajectories $\boldsymbol{\tau}_j = \{\mathbf{y}_t^j\}_{t=1..N_t}$, $j = 1..N_d$, by obtaining the weights $\boldsymbol{\omega}_j$ of each demonstration through least squares. Subsequently, the parameters of the distribution $\boldsymbol{\theta} = \{\boldsymbol{\mu}_\omega, \mathbf{\Sigma}_\omega, \mathbf{\Sigma}_y\}$, $\mathbf{\Sigma}_y$ being the state covariance, are fitted by means of a maximum likelihood estimate, i.e., computing the sample mean and the sample covariance of $\boldsymbol{\omega}$. Then the probability of observing a $\mathbf{y}_t$ is:

$$\begin{aligned} p(\mathbf{y}_t;\boldsymbol{\theta}) &= \int \mathcal{N}(\mathbf{y}_t|\mathbf{\Phi}_t^T\boldsymbol{\omega}, \mathbf{\Sigma}_y)\mathcal{N}(\boldsymbol{\omega}|\boldsymbol{\mu}_\omega, \mathbf{\Sigma}_\omega)d\boldsymbol{\omega} \\ &= \mathcal{N}\left(\mathbf{y}_t|\mathbf{\Phi}_t^T\boldsymbol{\mu}_\omega, \mathbf{\Sigma}_y + \mathbf{\Phi}_t^T\mathbf{\Sigma}_\omega\mathbf{\Phi}_t\right) \end{aligned} \tag{3}$$

Due to the probabilistic representation, ProMPs can represent motion variability while keeping other MP properties such as rescalation and linear representation wrt. parameters. ProMPs also allow for other operations such as modulation via probabilistic conditioning and combination by product, as well as providing a model-based stochastic controller that reproduces the encoded trajectory distribution [11].

A. Dimensionality Reduction of ProMPs

In [1], the authors proposed to reduce the dimensionality of a ProMP by performing linear DR in the space of degrees of freedom (DoF) of the robot, to reduce the number of DoFs from d to r. This had the impact of reducing the dimensionality of the parameter vector $\boldsymbol{\omega}$ from dN_f to rN_f, with $r < d$, N_f being the number of Gaussian kernels used per DoF. While reducing the dimensionality in the robots' DoF has advantages such as a better qualitative understanding of the task, a smaller linear projection matrix which is easier to estimate, and it is also used in other approaches [18]; However, here we propose to reduce the dimensionality in the space of the Gaussian weight vectors $\boldsymbol{\omega}$. This variation is introduced to then build a GMM in the common latent parameter space and has the advantage of fine-tuning the dimensionality of the latent space, given that we can encode actions that are more different, without loosing too much information. In this section, we describe the expression of such DR, which is derived in the Appendix. We will reduce the dimensionality through a linear mapping $\mathbf{\Omega}$ ($dN_f \times M_f$) so that the state is represented as:

$$\mathbf{y}_t = \mathbf{\Psi}_t^T\mathbf{\Omega}\boldsymbol{\nu} + \boldsymbol{\epsilon}_y, \tag{4}$$

where $\boldsymbol{\nu}$ is an M_f-dimensional weight vector in the latent space of $\boldsymbol{\omega}$. This results in a new probability distribution of $\mathbf{y}_t$ (compared to Eq.(3)):

$$\begin{aligned} p(\mathbf{y}_t;\boldsymbol{\theta}) &= \int \mathcal{N}(\mathbf{y}_t|\mathbf{\Phi}_t^T\mathbf{\Omega}\boldsymbol{\nu}, \mathbf{\Sigma}_y)\mathcal{N}(\boldsymbol{\nu}|\boldsymbol{\mu}_\nu, \mathbf{\Sigma}_\nu)d\boldsymbol{\nu} \\ &= \mathcal{N}\left(\mathbf{y}_t|\mathbf{\Phi}_t^T\mathbf{\Omega}\boldsymbol{\mu}_\nu, \mathbf{\Sigma}_y + \mathbf{\Phi}_t^T\mathbf{\Omega}\mathbf{\Sigma}_\nu\mathbf{\Omega}^T\mathbf{\Phi}_t\right) \end{aligned} \tag{5}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\mu}_\nu, \mathbf{\Sigma}_\nu, \mathbf{\Omega}, \mathbf{\Sigma}_y\}$ is the set of parameters of this DR-ProMP representation. In order to obtain such parameters, two main approaches can be used, as in [1]: Weighted Maximum likelihood estimation (WMLE) -derived in the Appendix-, or Principal Component Analysis (PCA), which is also helpful to obtain a sufficient value for M_f and an initialization to the problem.

III. Gaussian Mixture Models (GMMs) and Regression (GMR)

In this section, we recall the basic expressions of GMMs and GMR, that will be used together with a reward-weighted responsibility as detailed in Section III-C.

A. Gaussian Mixture Models

A Gaussian Mixture Model (GMM) distribution over a random variable $\mathbf{x}$ can be written as a weighted superposition of K Gaussians with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$, weighted by their mixing proportions π_k [12], for $k = 1..K$:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{6}$$

where it is common to consider π_k as the probability of a K-dimensional random variable $\mathbf{z}$, with $z_k \in \{0, 1\}$ and $p(z_k = 1) = p(z_k = 1, z_{\neq k} = 0) = \pi_k$ [12]. Note that $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^{K} \pi_k = 1$. Therefore, $p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ and, marginalizing $\mathbf{x}$ wrt. $\mathbf{z}$ we obtain Eq. (6) again:

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{7}$$

The GMM model in Eq.(6) can be obtained with an Expectation-Maximization (EM) algorithm [12], for which we need to compute a term $\gamma(z_{ik}) \triangleq p(z_k = 1|\mathbf{x}_i)$ by using the Bayes' rule:

$$\gamma(z_{ik}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \tag{8}$$

$\gamma(z_{ik})$ is called the *responsibility* of the component k associated with a sample $\mathbf{x}_i$. Using the aforementioned EM algorithm with a set of N data samples $\mathbf{X}$, the log-likelihood of the GMM model and the data $\ln p(\mathbf{X}|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ can be maximized in an iterative way:

- **E-step**: Evaluate the responsibilities using current parameters with Eq. (8).
- **M-step**: Re-calculate the parameters with (see [12]):

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{ik}) \mathbf{x}_i \tag{9}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{ik})(\mathbf{x}_i - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_i - \boldsymbol{\mu}_k^{\text{new}})^T \tag{10}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \tag{11}$$

where $N = \sum_k N_k$ and

$$N_k = \sum_{i=1}^{N} \gamma(z_{ik}) \tag{12}$$

- **Evaluate** the log-likelihood:

$$\ln p(\mathbf{X}|\pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{N} \ln \left[\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right], \tag{13}$$

and check for convergence.

An important point when using GMMs is to decide the number of Gaussians that will form the model, i.e.: K. In case there is no prior knowledge, the number of components is often set arbitrarily or using clustering methods. In this work, we used Persistent Homology [13], which permits computing the *Betti numbers* of the triangulation of a data set, to determine the number of clusters depending on a threshold. This threshold defines the maximum distance allowed to connect two data points when building the triangulation and, using computational topology tools [13], we can evaluate the evolution of the number connected components (0-Betti number), one-dimensional holes (1-Betti number) and so on wrt. the selected threshold. In the case of GMM, the 0-Betti number provides useful information to decide on the number of Gaussian components. The choice of the maximum threshold is crucial for obtaining both valid results and computationally inexpensive calculations. We found empirically that, for the different datasets we tried, a good threshold was $L_{\max} = \frac{1}{2}\sqrt{N_d \max_i \lambda_i}$, where λ_i are the eigenvalues of the covariance matrix on the dataset of vectors $\mathbf{x}$.

B. Gaussian Mixture Regression

Gaussian Mixture Regression (GMR)[2] is a regression tool that, given a GMM on a variable $\mathbf{x} = \begin{bmatrix} \mathbf{s} \\ \boldsymbol{\nu} \end{bmatrix}$ (note we already use the notation for the contextual variables $\mathbf{s}$ and latent space ProMP weights $\boldsymbol{\nu}$), approximates the probability $p(\boldsymbol{\nu}|\mathbf{s})$. Given the GMM on $\mathbf{x}$:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\begin{array}{c|cc} \mathbf{s} & \boldsymbol{\mu}_k^s & \boldsymbol{\Sigma}_k^s \quad \boldsymbol{\Sigma}_k^{s\nu} \\ \boldsymbol{\nu} & \boldsymbol{\mu}_k^\nu & \boldsymbol{\Sigma}_k^{\nu s} \quad \boldsymbol{\Sigma}_k^\nu \end{array} \right), \tag{14}$$

this approximation is often calculated so that the resulting conditioned distribution is a unimodal Gaussian. However, in this work, we keep the result of the regression as a multimodal distribution, i.e.: a GMM on the variable $\mathbf{z}$, given a value of $\mathbf{y}$. This can be obtained by setting [2]:

$$p(\boldsymbol{\nu}|\mathbf{s}) \sim \sum_{k=1}^{K} h_k(\mathbf{s}) \mathcal{N}\left(\hat{\boldsymbol{\mu}}_k^\nu(\mathbf{s}), \hat{\boldsymbol{\Sigma}}_k^\nu \right), \tag{15}$$

where

$$\hat{\boldsymbol{\mu}}_k^\nu(\mathbf{s}) = \boldsymbol{\mu}_k^\nu + \boldsymbol{\Sigma}_k^{\nu s}(\boldsymbol{\Sigma}_k^s)^{-1}(\mathbf{s} - \boldsymbol{\mu}_k^s) \tag{16}$$

and

$$\hat{\boldsymbol{\Sigma}}_k^\nu = \boldsymbol{\Sigma}_k^\nu - \boldsymbol{\Sigma}_k^{\nu s}(\boldsymbol{\Sigma}_k^s)^{-1}\boldsymbol{\Sigma}_k^{s\nu} \tag{17}$$

and the mixing proportions are calculated as:

$$h_k(\mathbf{s}) = \frac{\pi_k \mathcal{N}(\mathbf{s}|\boldsymbol{\mu}_k^s\, \boldsymbol{\Sigma}_k^s)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{s}|\boldsymbol{\mu}_j^s\, \boldsymbol{\Sigma}_j^s)} \tag{18}$$

The rationale behind not approximating $p(\boldsymbol{\nu}|\mathbf{s})$ with a single Gaussian is to keep the shape of each one of the different actions associated with each component of the GMM. Using GMR allows to generate stochastic motion in the following manner: We can firstly observe a contextual variable $\mathbf{s}_{\text{sample}}$, and obtain the conditioned mixture model $p(\boldsymbol{\nu}|\mathbf{s})$. We can then generate a sample $\boldsymbol{\nu}_{\text{sample}}$ from such conditioned GMM, and map it to the robot state with Eq. (4) to obtain a sample that will be adapted to the observed contextual variable $\mathbf{s}$. The responsibilities associated to each action after conditioning will also provide enough information of which action -or cluster- is predominant for each context. Nevertheless, in case there actually exists an overlap between components, we also use a method for updating the GMM with a reward-weighted responsibility [14], as presented in subsection III-C.

C. Reward-Weighted Responsibility GMM

In the previous sections, we introduced an approach based on GMM and GMR which, using a joint distribution of the latent space ProMP parameters and context variables, proves to be very useful, for instance, as a generative model for providing trajectories given a context sample value. However, this model can be further improved by evaluating a sample and changing the model according to the behavior observed. Let $\mathbf{s}_j$ be a contextual value observed, and $\boldsymbol{\nu}_j$ a sample from the distribution of $\boldsymbol{\nu}$ given $\mathbf{s}_j$, as explained in the previous section. Then, let r_{jk} be a reward associated with sampling from the action -or GMM component- k given context sample $\mathbf{s}_j$. If the rewards r_{jk} of a number of rollouts $j = 1..N_s$ are converted into relative weights d_j using a policy search algorithm such as REPS [15], we can then define the *reward-weighted responsibility* of an action as

$$\varphi_{jk} = d_{jk}\gamma(z_{jk}), \tag{19}$$

Given such weighted responsibilities, we can substitute $d_{sk}\gamma(z_{sk})$ for $\gamma(z_{sk})$ in Eqs.(9)-(12) and obtain a Reward-Weighted Responsibility GMM (RWR-GMM) [14], which can be used both for selecting a ProMP component from the mixture and to adapt it according to the contextual variable by GMR III-B.

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k}\sum_{j=1}^{N} d_{jk}\gamma(z_{jk})\mathbf{x}_j \tag{20}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k}\sum_{j=1}^{N} d_{jk}\gamma(z_{jk})(\mathbf{x}_j - \boldsymbol{\mu}_k^{\text{new}})(\mathbf{x}_j - \boldsymbol{\mu}_k^{\text{new}})^T \tag{21}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N}, \tag{22}$$

where $N = \sum_k N_k$ and

$$N_k = \sum_{j=1}^{N} d_{jk}\gamma(z_{jk}). \tag{23}$$

Algorithm 1 RWR-GMM

Input: GMM^0 with $\{\pi_k^0, \boldsymbol{\Sigma}_k^0, \boldsymbol{\mu}_k^0, N_k^0\}$
Number of samples N_s per update. Number of updates N_{updates}
Output: Updated GMM parameters $\{\pi_k, \boldsymbol{\Sigma}_k, \boldsymbol{\mu}_k, N_k\}$

```
for epoch = 1 : N_updates do
    for j = 1 : N_s do
        Generate or observe a context value s_j
        Obtain sample ν_j given s_j with GMR (Eq. (15))
        Execute the motion ν_j and evaluate r_j
    end for
    Obtain relative importance weights d_j for each execution, given the rewards r_j.
    while no convergence do
        Update the GMM iterating through Eqs. (24)-(29) until convergence.
    end while
    Update the old distribution GMM^0=GMM for next epoch
end for
```

D. Iterative RWR-GMM

Moreover, given an initial GMM composed of $\{\mu_k^0, \Sigma_k^0, \pi_k^0, N_k^0\}_{k=1..K}$ we can add a set of N new samples $\{\mathbf{x}_1, ..., \mathbf{x}_N\} = \{\boldsymbol{\nu}_1, \mathbf{s}_1; ...; \boldsymbol{\nu}_N, \mathbf{s}_N\}$ with their associated rewards $r_1, ..., r_N$. In this work, we propose to update the model by iterating through equations (20)-(23), keeping in mind some of the terms corresponds to the previous model, and others correspond to the effect of the new samples. Otherwise, the effect of the old model would fade at every iteration. Then, for each iteration i, we compute:

$$\varphi_{jk}|_{i+1} = d_n \frac{\pi_k|_i \mathrm{p}\left(\mathbf{x}_n|\boldsymbol{\mu}_k|_i, \boldsymbol{\Sigma}_k|_i\right)}{\sum_k \pi_k|_i \mathrm{p}\left(\mathbf{x}_n|\boldsymbol{\mu}_k|_i, \boldsymbol{\Sigma}_k|_i\right)} \tag{24}$$

$$N_k|_{i+1} = N_k^0 + \sum_{j=1}^{N_s} \varphi_{jk}|_{i+1} \tag{25}$$

$$\pi_k|_{i+1} = \frac{N_k|_{i+1}}{\sum_{l=1}^{K} N_l|_{i+1}} \tag{26}$$

$$\boldsymbol{\mu}_k|_{i+1} = \frac{1}{N_k|_{i+1}}\left[N_k^0\boldsymbol{\mu}_k^0 + \sum_{j=1}^{N_s}\varphi_{jk}|_{i+1}\mathbf{x}_j\right] \tag{27}$$

$$\boldsymbol{\Sigma}_k|_{i+1} = \frac{N_k^0\boldsymbol{\Sigma}_k^0}{N_k|_{i+1}} + \\ + \frac{1}{N_k|_{i+1}}\left[\sum_{n=1}^{N_s}\varphi_{jk}|_{i+1}(\mathbf{x}_n - \boldsymbol{\mu}_k|_{i+1})(\mathbf{x}_n - \boldsymbol{\mu}_k|_{i+1})^T\right] \tag{28}$$

Additionally, a forgetting factor λ can be added to the update of N_k, so as to give more importance to new samples. This will allow new samples to have a higher impact on the

Algorithm 2 Building a GMM-DRProMP model

Input: $\{\mathbf{y}_t^j\}_{t,j}$, for each trajectory sample $j = 1..N_d$ and each timestep $t = 1..N_t$.
Output: GMM parameters $\{\pi_k, \mathbf{\Sigma}_k, \boldsymbol{\mu}_k\}$ and latent space projection $\mathbf{\Omega}$

1: Use all trajectory data $\{\mathbf{y}_t^j\}_{t,j}$ with PCA to initialize $\boldsymbol{\theta} = \{\boldsymbol{\mu}_\nu, \mathbf{\Sigma}_\nu, \mathbf{\Omega}, \mathbf{\Sigma}_y\}$, as well as the latent space dimension M_f.
2: Use data $\{\mathbf{y}_t^j\}_{t,j}$ and $\boldsymbol{\theta}$ to obtain a new $\mathbf{\Omega}$ with Eq.(38).
3: Obtain the Gaussian weight vector $\boldsymbol{\nu}_j$ for each trajectory $\{\mathbf{y}_t^j\}_t$.
4: Using the joint variable $\mathbf{x} = \begin{bmatrix} \mathbf{s} \\ \boldsymbol{\nu} \end{bmatrix}$, cluster data and determine the number of clusters using persistent homology.
5: Build the GMM on $\mathbf{x}$ using Eqs. (9)-(12).

results as, otherwise, N_k will grow every time the policy is updated and new samples will have less effect.

$$N_k = N_k \lambda \tag{29}$$

Algorithm 1 how a RWR-GMM is updated. For a number of epochs, N_s samples are generated by observing a context value $\mathbf{s}_j$ and conditioning the GMM to it. Once all the samples have been run in the robot/simulator, their evaluations (rewards) can be converted to relative weights with a policy search method as REPS [15]. Then, the GMM is updated by iterating Eqs. (24)-(29) until convergence.

IV. Method Overview

The proposed framework is summarized in Algorithm 2. In order to build a model, we firstly use Principal Component Analysis, without taking contextual variables into consideration, in order to initialize a ProMP with all the sample trajectories. This PCA will provide an initialization of the unimodal model that we will refine with the EM algorithm in the Appendix, but most importantly, provides a way of setting the dimensionality $M_f < dN_f$. In this work, we took M_f so that the first M_f singular values $\sigma_{1..M_f}$ represent a portion ρ of the information, i.e.:

$$\min M_f, \text{ s.t. } \frac{\sum_{i=1}^{M_f} \sigma_i^2}{\sum_{i=1}^{dN_f} \sigma_i^2} \geq \rho. \tag{30}$$

After initializing and refining $\mathbf{\Omega}$, we fit each trajectory j weight vector $\boldsymbol{\nu}_j$ by solving (through least squares):

$$\begin{bmatrix} \mathbf{y}_1^j \\ \dots \\ \mathbf{y}_{N_t}^j \end{bmatrix} = \begin{bmatrix} \mathbf{\Psi}_1 \\ \dots \\ \mathbf{\Psi}_{Nt} \end{bmatrix} \mathbf{\Omega} \boldsymbol{\nu}_j. \tag{31}$$

Once having all $\boldsymbol{\nu}_j$, $j = 1..N_d$, and their respective observed contextual values $\mathbf{s}_j$, we can use their joint variable $\mathbf{x}_j$ and, applying persistent homology (see Section III-A) obtain the number of clusters to build our model.

With the resulting GMM, we can generate trajectories given a contextual value $\mathbf{s}_j$ by using GMR as detailed in Sec. III-B. GMR will provide a new GMM where all actions -or GMM components- are conditioned to the observed context value. Then, the proportion weights $h_k(\mathbf{s}_j)$ in Eq. (18) will allow the GMM to select which component to use and therefore generate a sample $\boldsymbol{\nu}_n$, which can be translated into a robot trajectory by using $\mathbf{y}_t \simeq \mathbf{\Psi}_t^T \mathbf{\Omega} \nu_n$.

This trajectory can then be executed and evaluated, obtaining a reward for sample j and component k, r_{jk}. This reward can be converted to a normalized weight d_{jk} by suitable PS methods as REPS [15]. The resulting weight, together with the sample $\mathbf{x}_j = [\mathbf{s}_j; \boldsymbol{\nu}_j]$, can then be applied to update the RWR-GMM with Eqs. (20)-(23).

In the following section, we present three experiments showing the advantages of our proposed method.

V. Experimentation

A. First experiment: modelling a 2D letter dataset

As a first validation test, we used a dataset from [16], consisting of hand-written letters. We took the first 12 characters in the alphabet (*A, B, C, D, E, F, G, H, I, J, K, L*). The trajectories obtained lie in a 2-dimensional space, and we used $N_f = 10$ for each DoF, and set $\rho = 0.98$ in (30). We obtained $M_f = 14$ by using PCA on all the trajectory data, and then performed EM as detailed in the Appendix to obtain the latent space projection $\boldsymbol{\omega} \Longrightarrow^{\Omega} \boldsymbol{\nu}$. Next, we used the *Javaplex* library [17] to compute the barcode representing the variation in the 0-Betti number, which correctly hinted there were 12 components, as seen in Fig. 2. After setting

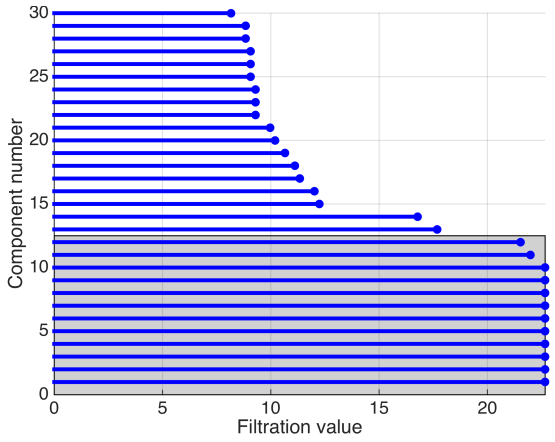

Fig. 2. Persistence of the 30 most important connected components in the data as the filtration value grows. For a filtration value (ball radius) of zero, each sample (158 trajectories in this case) is a connected component. When increasing the filtration value, different samples are linked and so, the number of connected components decreases. In this figure, we see that there are possibly 12 components in the data.

the number of components to $K = 12$, we fitted a GMM to the space composed of samples $\mathbf{x} = [\mathbf{s}; \boldsymbol{\nu}]$, $\mathbf{s}$ being 1 for the letter *A*, 2 for letter *B*, and so on.

With such a model, we could then condition the GMM to observations $\mathbf{s} = \{1, .., 12\}$ and obtain well-written characters, as seen in Fig. 3. Additionally, we tested how

the method would behave for intermediate values of $\mathbf{s}$, evaluating the mixing proportions of the GMM conditioned on $\mathbf{s}$ obtained through GMR, as displayed in Fig. 4. There, we can see there are parts in the $\mathbf{s}$ domain where all the mixing proportions of the conditioned model are zero. This means that, given the data, the model assigns a very small probability to all the actions (letters) and, therefore, selects none. The domains in Fig. 4 can be widened with a strong covariance regularization when fitting the GMM on $\mathbf{x}$ through EM, but this resulted in meaningless characters.

B. Second Experiment: Feeding a mannequin

As a second experiment, we scaled up the dimensionality of the problem with a feeding experiment (see Fig. 1). We placed a mannequin's head on a table, together with two plates, one with pieces of apple and another one with small balls simulating soup. With a Kinect camera reading the head and two plates' positions through QR codes attached to them, the aim of the experiment was to teach the robot how to feed the mannequin either of the two types of food (apple, soup). We kinesthetically taught the robot 20 motions for feeding each food (40 demonstrations in total), changing the position of the head and plates in every demonstration. For the robot state, we considered the pose of the end-effector (6-DoF), with 15 Gaussians per DoF, totalling 90 parameters, which were mapped onto a 27-dimensional latent space through our proposed DR. Regarding contextual representation, we considered the $x-y$ coordinates on the table of both plates and the head, plus a choice variable which was 1 for soup and 2 for apples. In total, $\dim(\boldsymbol{\nu}) = 27$ and $\dim(\mathbf{s}) = 7$, totaling a state $\mathbf{x} = \begin{bmatrix} \boldsymbol{\nu} \\ \mathbf{s} \end{bmatrix}$ of $\dim(\mathbf{x}) = 34$. In the video attachment (also found in `http://www.iri.upc.edu/groups/perception/#DRGMM`), a good qualitative behavior of the resulting model when conditioning to a given context is shown, see also Fig. 5.

C. Third experiment: modeling peg insertion in a movable hole

Last we used the proposed method for learning and improving a multiple-solution task and used Algorithm 1 to improve it over experience. We generated a 2-dimensional ProMP with 10 Gaussians per DoF. As input data, we generated planar trajectories aiming at a goal point in a semicircle (see Fig. 7), and also going through a via-point generated by adding and offset to the mid-point of the straight line connecting the origin and the goal point. We also perturbed the ending point of the trajectory to add error to the initial trajectories, and we took $\mathbf{s} = [s_1; s_2]$, where s_2 is the desired orientation of arrival to a hole placed at the goal point. The context variables were initialized with a GMM with means $\boldsymbol{\mu}_{s_1} = [\frac{\pi}{6}; \frac{2\pi}{5}; \frac{13\pi}{20}; \frac{11\pi}{14}]$, $\boldsymbol{\mu}_{s_2} = \boldsymbol{\mu}_{s_1} + \epsilon$ (ϵ being a zero-mean Gaussian value with variance 0.1) and covariances $\boldsymbol{\Sigma}_s = 0.05\mathbf{I}$ for all components. The reward function was then the sum of the error in reaching the goal point, plus a term penalizing the angle between the peg arrival velocity and the desired insertion orientation in the hole, also seen in Fig. 7. Furthermore, we added an acceleration term to prevent too-jerky trajectories.

We fitted the model applying Algorithm 2, and obtained a latent space dimension $M_f = 4$. By using persistent homology (similarly to Sec. V-A), we found the number of components to be also 4.

Given the initialized model, we performed a learning experiment using Algorithm 1 by sampling context from the original context distribution, and converting rewards to weights by using Relative Entropy Policy Search [15]. We did 100 policy updates with 100 samples each, using a decay factor of $\lambda = 0.8$ in Eq.(29). In Fig. 6, we can see the learning curve showing the average sample reward after every policy update (averaged through 10 learning trials). Our full proposed method (RWR+GMM+DR) is compared against a unimodal case with DR (RWR+DR), a multimodal model without DR (RWR+GMM) and contextual REPS with covariance matrix adaptation [5]. The results show that a multimodal model is truly preferable and, while adding DR might generate a lower-reward in the early stages due to having a smaller model, DR allows to learn the task faster. Figure 7 displays some examples of the final policy.

VI. Conclusions

In this paper, we proposed a generative model that can store a complex set of actions that share a common contextual feature dependency. The proposed method uses a latent space projection of the MP parameters of each action and is capable of drastically reducing the dimensionality of the problem and, thus the complexity of both action selection and adaptivity. Moreover, we proposed a tool based on persistent homology to assess how many actions -Gaussian components- we must consider given a dataset by using Persistent Homology, and we derived the equations required to improve the model when new experiences through an adapted EM algorithm. The results show that the proposed method is not only capable of fitting a lower-dimensional contextualized GMM, but also to improve it with the gained experience using direct policy search reinforcement learning methods. Future work includes assessing the capability of the framework to adapt to changing context in an online manner, which can be done by blending the currently reproduced trajectory with a newly adapted one, and also to perform DR in each action separately, with a common parameter vector for all components of the GMM.

Appendix: DR-ProMPs equations

A. Fitting DR-ProMP parameters with EM

In [1], the authors devised an Expectation-Maximization based method for finding a linear Dimensionality Reduction (DR) technique to represent ProMPs in a latent space of the DoF of the robot, using the expression $\mathbf{y}_t = \boldsymbol{\Omega}\boldsymbol{\Psi}_t^T\boldsymbol{\omega}$ for the state space, where $\boldsymbol{\Omega}$ was a $(d \times r)$ matrix, d being the DoF of the trajectory and r a latent space dimension. In this paper, we used the reverse order between the two terms $\boldsymbol{\Omega}$ and $\boldsymbol{\Psi}_t$, resulting in:

$$\mathbf{y}_t = \boldsymbol{\Psi}_t^T\boldsymbol{\Omega}\boldsymbol{\nu}, \tag{32}$$

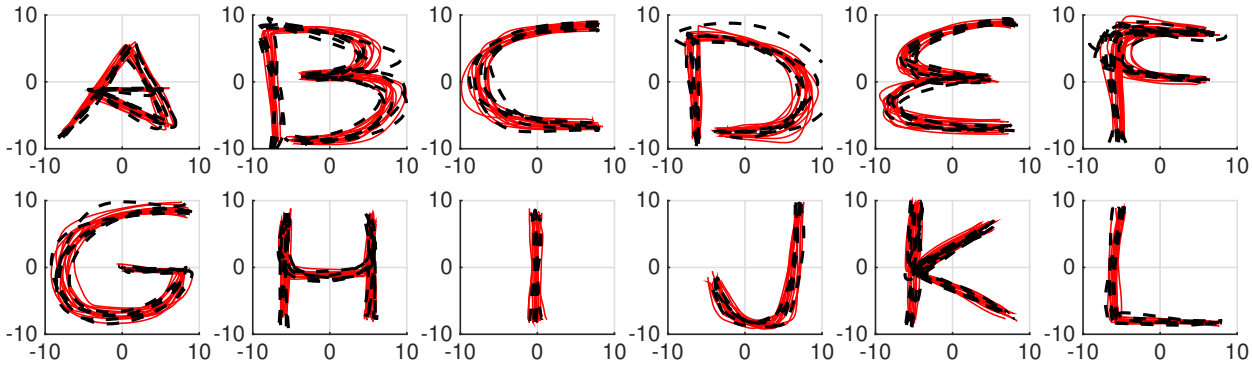

Fig. 3. Re-generated letters. Dataset (in red) and different samples generated by conditioning to $\mathbf{s} = \{1, .., 12\}$ (in discontinuous black).

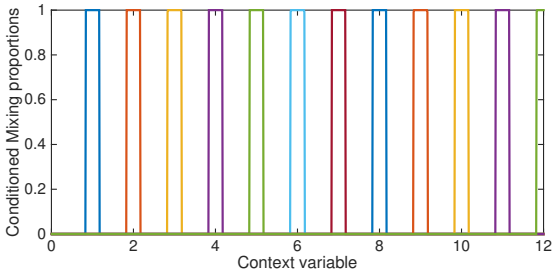

Fig. 4. Mixing proportions in the regenerating letter experiment.

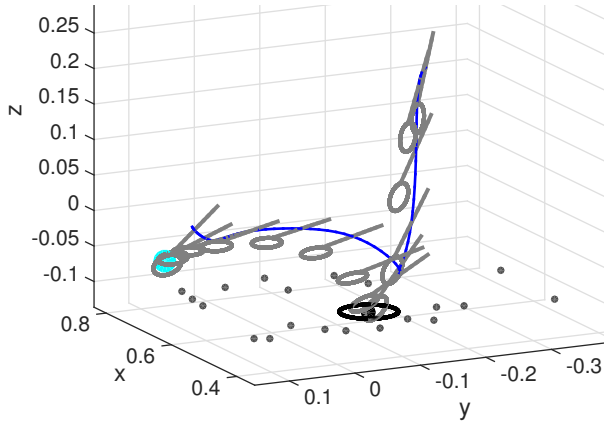

Fig. 5. Conditioning the obtained model. Given a soup query, the corresponding plate is localized (in black) and a motion (in blue with the spoon drawn in grey) is performed to feed the mouth (in cyan). The samples used for this action are also shown as dots in dark grey.

where $\mathbf{\Omega}$ is now a $(dN_f \times M_f)$ matrix and we use $\boldsymbol{\nu} \sim \mathcal{N}(\boldsymbol{\mu}_\nu, \mathbf{\Sigma}_\nu)$ to represent the latent space parameters. In this case, the linear DR is performed directly in the Gaussian weights space of the ProMP, instead of the DoF' space, resulting in the probability distribution for the state space:

$$\begin{aligned} p(\mathbf{y}_t) &= \int_{\boldsymbol{\nu}} \mathcal{N}\left(\mathbf{y}_t | \mathbf{\Psi}_t^T \mathbf{\Omega} \boldsymbol{\nu}, \mathbf{\Sigma}_y\right) \mathcal{N}\left(\boldsymbol{\mu}_\nu, \mathbf{\Sigma}_\nu\right) d\boldsymbol{\nu} \\ &= \mathcal{N}\left(\mathbf{y}_t | \mathbf{\Psi}_t^T \mathbf{\Omega} \boldsymbol{\mu}_\nu, \mathbf{\Sigma}_y + \mathbf{\Psi}_t^T \mathbf{\Omega} \mathbf{\Sigma}_\nu \mathbf{\Omega} \mathbf{\Psi}_t^T\right) \end{aligned} \tag{33}$$

We now describe the Expectation-Maximization step for obtaining the parameters $\boldsymbol{\theta} = \{\boldsymbol{\mu}_\nu, \mathbf{\Sigma}_\nu, \mathbf{\Omega}, \mathbf{\Sigma}_y\}$ by using the

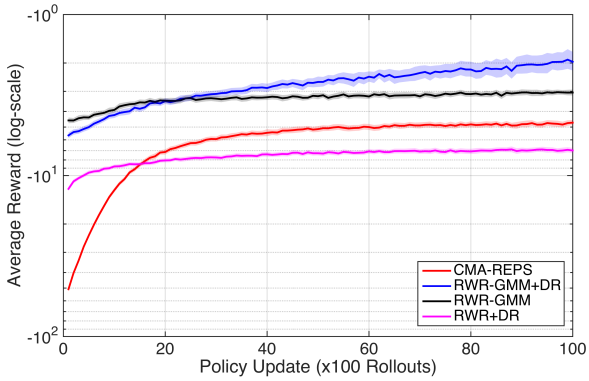

Fig. 6. Learning curves obtained with four methods in the peg-in-hole experiment. Mean rewards and their 95% confidence intervals obtained after 100 experiments of 100 policy updates with 100 samples each.

complete-data log-likelihood [12]. $\mathbf{\Psi}_t^T$ will now be a $(d \times M_f)$ matrix, and $\mathbf{y}_t^j$ will be a d-dimensional vector. We will use the vector $\mathbf{Y}^j$ to denote the concatenated position vectors of a single trajectory j,

$$\mathbf{Y}^j = \left[\mathbf{y}_1^{jT}, ..., \mathbf{y}_{N_t}^{j\ T}\right]^T.$$

Similarly to [1], we want to use our model estimation algorithm for policy search algorithms that are based on data reweighting. These algorithms introduce a weighting d_k for each trajectory. Hence, we also have to consider such a weighting in our EM algorithm. We will maximize the weighted marginal log-likelihood $\sum_k d_k \log p(\mathbf{y}_t | s, \boldsymbol{\theta})$ of the data and the latent space representation, thus we have to derive the equations with the marginalized $\boldsymbol{\nu}$ with the difficulties it entails. The following subsections explain how to obtain the log-likelihood function and differentiate it.

1) Expectation step

In the expectation step, we must find the probabilities for each demonstration k with the old parameters $\boldsymbol{\theta}^{old}$:

$$p(\boldsymbol{\nu}|\mathbf{Y}^j) = \frac{p(\mathbf{Y}^j|\boldsymbol{\nu})p(\boldsymbol{\nu})}{p(\mathbf{Y}^j)} \propto p(\mathbf{Y}^j|\boldsymbol{\nu})p(\boldsymbol{\nu}), \tag{34}$$

where:

$$p(\mathbf{Y}^j|\boldsymbol{\nu}) = \mathcal{N}(\mathbf{Y}^j|\mathbf{\Psi}^T\mathbf{\Omega}\boldsymbol{\nu}, \mathbf{I}_{N_t} \otimes \mathbf{\Sigma}_y). \tag{35}$$

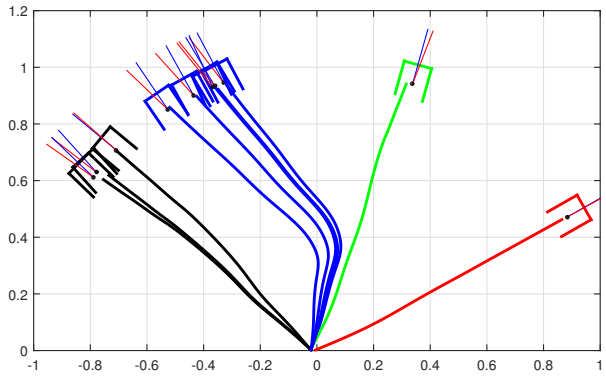

Fig. 7. Examples of trajectories generated after learning (x and y components). Each one of the 4 actions found is plotted in a different color, with the orientated peg drawn in the end-effector's position. All the obtained trajectories after conditioning the obtained model to the context $\mathbf{s}$ are capable of achieving a low error on the end-point, with low orientation error as well, represented by the small segments on the trajectories end-points.

Note that in the contextual case, we have omitted the conditioning on the context variables for simplicity of the equations. Using the Bayes rule for Gaussian distributions, we obtain $p(\boldsymbol{\omega}|\mathbf{Y}^j) = \mathcal{N}(\boldsymbol{\nu}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$, where

$$\boldsymbol{\mu}_j = \boldsymbol{\mu}_\nu + \mathbf{K}_\nu \mathbf{s}_j + \boldsymbol{\Sigma}_\nu \boldsymbol{\Omega}_{N_t}^T \boldsymbol{\Psi} \boldsymbol{\Gamma}^{-1} (\mathbf{Y}^j - \boldsymbol{\Psi}^T \boldsymbol{\Omega}_{N_t} \boldsymbol{\mu}_\nu)$$

$$\boldsymbol{\Sigma}_j = \boldsymbol{\Sigma}_\nu - \boldsymbol{\Sigma}_\nu \boldsymbol{\Omega}_{N_t}^T \boldsymbol{\Psi} \boldsymbol{\Gamma}^{-1} \boldsymbol{\Psi}^T \boldsymbol{\Omega}_{N_t} \boldsymbol{\Sigma}_\omega,$$

and $\boldsymbol{\Gamma}$ is given by $\boldsymbol{\Gamma} = \mathbf{I}_{N_t} \otimes \boldsymbol{\Sigma}_y + \boldsymbol{\Psi}^T \boldsymbol{\Omega}_{N_t} \boldsymbol{\Sigma}_\nu \boldsymbol{\Omega}_{N_t}^T \boldsymbol{\Psi}$.

2) Maximization step

Given the probabilities $p(\boldsymbol{\omega}|\mathbf{Y}^j)$ for each demonstration, we now maximize the weighted expectation of the log-likelihood function, where d_j is used as weight for each trajectory, i.e.,

$$L = \sum_{j=1}^{N_d} d_j \mathbb{E}_{\boldsymbol{\nu}|\mathbf{Y}^j;\boldsymbol{\theta}^{old}} \left[\log\left(p(\boldsymbol{\nu}, \mathbf{Y}^j; \boldsymbol{\theta})\right)\right]$$

After some calculations, analogously to [1], we obtain:

$$\begin{aligned} L = &-\tfrac{1}{2}\left[\left(\sum_{j=1}^{N_d} d_j\right) \log|2\pi\boldsymbol{\Sigma}_\nu| + N_t \log|2\pi\boldsymbol{\Sigma}_y|\right] \\ &-\tfrac{1}{2}\sum_{j=1}^{N_d} d_j (\boldsymbol{\mu}_\nu - \boldsymbol{\mu}_j)\boldsymbol{\Sigma}_\nu^{-1}(\boldsymbol{\mu}_\nu - \boldsymbol{\mu}_j) \\ &-\tfrac{1}{2}\sum_{j=1}^{N_d} d_j \sum_{t=1}^{N_t} \left[(y_t^j - \boldsymbol{\Psi}_t^T \boldsymbol{\Omega} \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_y^{-1} (y_t^j - \boldsymbol{\Psi}_t^T \boldsymbol{\Omega} \boldsymbol{\mu}_j)\right. \\ &\left.+\mathrm{tr}(\boldsymbol{\Omega}^T \boldsymbol{\Psi}_t \boldsymbol{\Sigma}_y^{-1} \boldsymbol{\Psi}_t^T \boldsymbol{\Omega} \boldsymbol{\Sigma}_j)\right] - \tfrac{1}{2}\sum_{j=1}^{N_d} d_j \mathrm{tr}(\boldsymbol{\Sigma}_\nu^{-1}\boldsymbol{\Sigma}_j). \end{aligned}$$

Then, we derivate L w.r.t. $\boldsymbol{\theta} = \{\boldsymbol{\mu}_\nu, \boldsymbol{\Sigma}_\nu, \boldsymbol{\Omega}, \boldsymbol{\Sigma}_y^{-1}\}$:

$$\boldsymbol{\mu}_\nu = \left(\sum_{j=1}^{N_d} d_j\right)^{-1} \sum_{j=1}^{N_d} d_j(\boldsymbol{\mu}_j), \tag{36}$$

$$\boldsymbol{\Sigma}_\nu = \left(\sum_{j=1}^{N_d} d_j\right)^{-1} \sum_{j=1}^{N_d} d_j \left[\boldsymbol{\Sigma}_j + (\boldsymbol{\mu}_\omega - \boldsymbol{\mu}_j)(\boldsymbol{\mu}_\omega - \boldsymbol{\mu}_j)^T\right], \tag{37}$$

$$\begin{aligned} \boldsymbol{\Omega} = &\left[\sum_{t=1}^{N_t} d_j \boldsymbol{\Psi}_t^T \boldsymbol{\Sigma}_y^{-1} \boldsymbol{\Psi}_t\right]^\dagger \left[\sum_{t=1}^{N_t} \boldsymbol{\Psi}_t \boldsymbol{\Sigma}_y^{-1} \sum_{j=1}^{N_d} d_j y_t^j \boldsymbol{\mu}_j^T\right] \\ &\cdot \left[\sum_{j=1}^{N_d} d_j \left(\boldsymbol{\Sigma}_j + \boldsymbol{\mu}_j \boldsymbol{\mu}_j^T\right)\right]^\dagger \end{aligned} \tag{38}$$

$$\begin{aligned} \boldsymbol{\Sigma}_y = &\left(\sum_{j=1}^{N_d} d_j\right)^{-1} \tfrac{1}{N_t} \sum_{j=1}^{N_d}\sum_{t=1}^{N_t} d_j \left[\mathbf{y}_t^j (\mathbf{y}_t^j - \boldsymbol{\Psi}_t^T \boldsymbol{\Omega} \boldsymbol{\mu}_j)^T + \boldsymbol{\Psi}_t^T \right. \\ &\left. + \boldsymbol{\Omega}\boldsymbol{\mu}_j (\boldsymbol{\Psi}_t^T \boldsymbol{\Omega} \boldsymbol{\mu}_j - \mathbf{y}_t^j)^T + \boldsymbol{\Psi}_t^T \boldsymbol{\Omega} \boldsymbol{\Sigma}_j \boldsymbol{\Omega} \boldsymbol{\Psi}_t^T\right] \end{aligned} \tag{39}$$

REFERENCES

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

[2] H. Sung, "Gaussian mixture regression and classification," PhD thesis, Rice University, Houston, Texas, 2004.

[3] 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.

[4] A. Kupcsik, M. Deisenroth, J. Peters and G. Neumann, "Data-Efficient Generalization of Robot Skills with Contextual Policy Search". *AAAI Conf. on Artificial Intelligence*, pp. 1401-1407, 2013.

[5] A. Abdolmaleki, D. Simões, N. Lau, L. P. Reis and G. Neumann, "Contextual Relative Entropy Policy Search with Covariance Matrix Adaptation," *2016 Int. Conf. on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 94-99, 2016.

[6] N. Hansen, S. Muller, and P. Koumoutsakos, "Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES)." *Evolutionary Computation*, vol.11, no.1, pp.1-18,2003.

[7] C. Daniel, G. Neumann,O. Kroemer, and J Peters, "Hierarchical Relative Entropy Policy Search", *J. of Machine Learning Research (JMLR)*, no. 17, pp.1-50, 2016.

[8] E, Pignat and S. Calinon, "Learning adaptive dressing assistance from human demonstration", *Robotics and Autonomous Systems*, vol 93, pp 61-75, 2017.

[9] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," *IEEE/RSJ Int. Conf. on Intelligent Robots (IROS)*, pp. 365-371, 2011.

[10] N. Sommer, K. Kronander and A. Billard. "Learning Externally Modulated Dynamical Systems.", *IEEE/RSJ Int. Conf. on Intelligent Robots (IROS)*, 2017.

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

[12] C. Bishop. Pattern recognition and machine learning. Springer, 2007.

[13] H. Edelsbrunner, "Computational Topology: An Introduction". American Mathematical Society; New ed. edition, 2009.

[14] A. Colomé, S. Foix, G. Alenyà, and C. Torras, "Reward-weighted GMM and its application to action-selection in robotized shoe dressing". *ROBOT'2017: Third Iberian Robotics Conference*, 2017.

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

[16] S. Calinon, "A Tutorial on Task-Parameterized Movement Learning and Retrieval", *Intelligent Service Robotics*, Springer Berlin Heidelberg, no. 1, vol. 9, pp 1-29, 2016.

[17] A. Tausz, M. Vejdemo-Johansson, and H. Adams, "JavaPlex: A research software package for persistent (co)homology". *Int. Conf. on Mathematical Software*, pp. 129-136, 2014.

[18] 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.