Robot Learning

Winter Semester 2020/2021, Homework 4

Prof. Dr. J. Peters, J. Watson, J. Carvalho, J. Urain and T. Dam

TECHNISCHE UNIVERSITÄT DARMSTADT

Total points: 38
Due date: Monday, 01 February 2021 (23:59)

Name, Surname, ID Number

Problem 4.1 Trajectory Generation with Dynamical Systems [38 Points]

In this exercise we will use the Dynamic Motor Primitives (DMPs), described by the following dynamical system,

$$\ddot{y} = \tau^2 (\alpha (\beta (g - y) - (\dot{y}/\tau)) + f_w(z)), \tag{1}$$
$$\dot{z} = -\tau \alpha_z z, \tag{2}$$

where y is the state of the system, $\dot{y}$ and $\ddot{y}$ are the first and second time derivatives, respectively. The attractor's goal is denoted by g and the forcing function by f_w. The parameters α and β control the spring-damper system. The phase variable is denoted by z and the temporal scaling coefficient by τ. The forcing function f_w is given by

$$f_w(z) = \frac{\sum_{i=0}^{K} \phi_i(z) w_i z}{\sum_{j=0}^{K} \phi_j(z)} = \boldsymbol{\psi}(z)^T \boldsymbol{w}, \quad \text{with} \quad \psi_i(z) = \frac{\phi_i(z) z}{\sum_{j=1}^{K} \phi_j(z)}, \tag{3}$$

where the basis functions $\phi_i(z)$ are Gaussian basis given by

$$\phi_i(z) = \exp\left(-0.5 (z - c_i)^2 / h_i\right), \tag{4}$$

where the centers c are equally distributed in the phase z, and the width h is an open parameter. For the programming exercises a basic environment of a double link pendulum is provided, as well as the computation of the $\boldsymbol{\psi}_i(z)$.

a) Similarities to a PD controller [2 Points]

Transform Equation (1) to have a similar structure to a PD-controller,

$$\ddot{y}_z = K_P \left(y_z^{des} - y_z\right) + K_D \left(\dot{y}_z^{des} - \dot{y}_z\right) + u_{ff} \tag{5}$$

and write down how the following quantities K_p, K_d, y_z^{des} and $\dot{y}_z^{des}$ look like in terms of the DMP parameters. Do not expand the forcing function $f_w(z)$ at your solutions.

b) Stability [2 Points]

Explain why the DMPs are stable when $t \to \infty$ and what would the equilibrium point be.

c) Double Pendulum - Training [12 Points]

Implement the DMPs and test them on the double pendulum environment. In order to train the DMPs you have to solve Equation (1) on the forcing function. Before starting the execution, set the goal g position to be the same as in the demonstration. Then, set the parameters to $\alpha = 25, \beta = 6.25, \alpha_z = 3/T, \tau = 1$. Use $N = 50$ basis functions, equally distributed in z. Use the learned DMPs to control the robot and plot in the same figure both the demonstrated trajectory and the reproduction from the DMPs. You need to implement the DMP-based controller (`dmpCtl.py`) and the training function for the controller parameters (`dmpTrain.py`). To plot your results you can use `dmpComparison.py`. Refer to `example.py` to see how to call it.

d) Double Pendulum - Conditioning on the Final Position [3 Points]

Using the trained DMPs from the previous question, simulate the system with different goal positions: first with $g_{t=\text{end}} = \{0, 0.2\}$ and then with $g_{t=\text{end}} = \{0.8, 0.5\}$. Generate one figure per DoF. In each figure, plot the demonstrated trajectory and the reproduced trajectories with different goal positions. How do you interpret the result?

e) Double Pendulum - Temporal Modulation [3 Points]

Using the trained DMPs from the previous question, simulate the system with different temporal scaling factors $\tau = \{0.5, 1.5\}$. Generate one figure per DoF and explain the result.

f) Probabilistic Movement Primitives - Radial Basis Function [3 Points]

We now want to use ProMPs. Before we train them, we need to define some basis functions. We decide to use $N = 30$ radial basis functions (RBFs) with centers uniformly distributed in the time interval $[0 - 2b, T + 2b]$, where T is the end time of the demonstrations. The bandwidth of the Gaussian basis (std) is set to $b = 0.2$. Implement these basis functions in `getProMPBasis.py`. Do not forget to normalize the basis such at every time-point they sum-up to one! Attach a plot showing the basis functions in time.

g) Probabilistic Movement Primitives - Training [7 Points]

In this exercise you will train the ProMPs using the imitation learning data from `getImitationData.py` and the RBFs defined in the previous question. Modify the `proMP.py` in order to estimate weight vectors w_i reproducing the different demonstrations. Then, fit a Gaussian using all the weight vectors. Generate a plot showing the desired trajectory distribution in time (mean and std) as well as the trajectories used for imitation.

h) Probabilistic Movement Primitives - Number of Basis Functions [2 Points]

Evaluate the effects of using a reduced number of RBFs. Generate two plots showing the desired trajectory distribution and the trajectories used for imitation as in the previous exercise, but this time use $N = 20$ and $N = 10$ basis functions. Briefly analyze your results.

i) Probabilistic Movement Primitives - Conditioning [4 Points]

Using Gaussian conditioning calculate the new distribution over the weight vectors w_i such as the trajectory has a via point at position $y^* = 3$ at time $t_{\text{cond}} = 1150$ with variance $\Sigma_{y^*} = 0.0002$. Use again 30 basis functions. Assuming that the probability over the weights is given by $\mathcal{N}\left(\boldsymbol{w}|\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\right)$ and the probability of being to that position is given by $\mathcal{N}\left(y^*|\boldsymbol{\Phi}\boldsymbol{w}, \Sigma_{y^*}\right)$, show how the new distribution over $\boldsymbol{w}$ is computed (how does the mean and variance look like)?
Then, in a single plot, show the previous distribution (learned from imitation) and the new distribution (after conditioning). Additionally, sample $K = 10$ random weight vectors from the ProMP, compute the trajectories and plot them in the same plot. Analyze briefly your results.