Advanced Robotics
Module on Probabilistic Movement Primitives

Sariah Mghames

[1]University of Lincoln

1. Exercise 1: Linear Regression

Task0- You are asked to generate random y data in the $init()$ function of the script $linear-regression.py$, (see 'Task0' in the script)

Task1- Complete the regression model: query a fit to the trained data (see 'Task1' in the script)

$y_{pred} = ...$

Task2- Plot the random data with their fitted curve (see 'Task2' in the script)

Task3- Query a prediction for a new input using the prediction model and print the output (see 'Task3 &4' in the script)

Task5- Change 'eta' variable which is the learning parameter and visualize the difference in the outcome

2. Exercise 2: Learn a ProMP from demonstrations

The main script is the ProPrimitive.py
It makes call for methods defined in promps.py, phase.py and basis.py.
The function that learns a ProMP is the following, inside 'ProPrimitive.py':
learner1.learnFromXDataTaskSapce(Q[:,:,0]/1000, tff1)
for each of the dimensions: X, Y and Z
for that you need to complete the following methods:
learnFromXDataTaskSpace()

learnFromYDataTaskSpace(),

learnFromZDataTaskSpace(),

in the promps.py script, and report on:

weightMatrix = np.zeros(... , ...)

temp = basisMatrix.transpose().dot(basisMatrix) + ...

weightVector = np.linalg.solve(temp, ...)

$$\mu^*_\omega = \mu_\omega + \Sigma_\omega \Psi_t (\Sigma^*_y + \Psi_t^T \Sigma_\omega \Psi_t)^{-1} (y^*_t - \Psi_t^T \mu_\omega)$$

$$\Sigma^*_\omega = \Sigma_\omega - \Sigma_\omega \Psi_t (\Sigma^*_y + \Psi_t^T \Sigma_\omega \Psi_t)^{-1} \Psi_t^T \Sigma_\omega$$

Figure 1. Conditioning

self.proMP.muX = np.mean(....... , axis=0)

Once you did this for one dimension you fill the same variables in the other dimensions

For that you need to refer to the maximum likelihood estimation solution given in the lecture by: $w = (\lambda I + \psi^T \psi)^{-1} \psi^T t$

To Complete: in order to be able to visualize the learnt ProMP through $traj1_Xee = learnedProMP1.getTrajectorySamplesX(....)$ that you can see in the $linear-regression.py$ script, you need to complete the section in the methods getTrajectorSamplesX(), getTrajectorySamplesY(), getTrajectorySamplesZ() in the promps.py script and report on the following:

$weightsX = np.random.multivariate - normal(...., ..., ...)$

trajectoryFlatX = ...

(same for each dimension)
Complete therefore the plot: ax.scatter(.... , ... , ... , c='b', marker='.') in the plot of title: 'Task space learnt promp1'

3. Exercise 3: Condition the learnt ProMP at a different Goal pose

In the given example the pose is constrained to only positions
Begin by inserting a goal position (X, Y, Z) from your choice
the function that do the conditioning is the following:
$traj - conditioned - tf = learnedProMP1.taskSpaceConditioning(...., ..., ...)$

fill in the gap and for the whole function to work you need to complete the sections:

LX = np.linalg.solve(desiredXVar[0,0] + ...dot(tempX), ...)

newProMP.muX = ... + LX.dot(desiredXmean[0] - basisMatrix.dot(....))

newProMP.covMatX = ... - LY.dot(basisMatrix).dot(.....)

in the promps.py script (same for each dimension) for that you can refer to the equations in figure 1.

4. Exercise 4: Play with the number of basis functions and their width and comment out on the observations

Make changes in the following:

basisGenerator = basis.NormalizedRBFBasisGenerator()

Comment: ....