Intelligent Control Systems Spring 2018 – Homework 3

Due at 11:00 p.m. on February 28, 2018
100 points

This is a Matlab homework assignment. You will create a files called "YOURLASTNAMEhw3p1.m" and 'YOURLASTNAMEhw3p2.m" and submit those files on Blackboard. Your file will do everything described in the problem statement on the next page. Do not submit any other .m files. Also copy the text in your .m file into a .docx or .pdf file and submit this file that is readable by Blackboard's plagarism software. The grading will be as follows:

- 100 points = correct solution
- 75 points = mostly correct solution
- 50 points = a good attempt, but significant improvements needed
- 25 points = a seriously flawed submission
- 0 points = no submission or failure to follow the submission instructions above

If you submit your homework on time or within your remaining late homework days, you will be given the chance to resubmit your solution if you did not get a score of 100. You will receive feedback and will need to resubmit by the deadline for the next homework assignment.

Please respect the rules on academic misconduct described in the course syllabus.

Problem 1:

You are going to use locally weighted regression to control a two-link pendulum. Each torque input for the pendulum is represented by the local model

$$y = w^\top \phi(\mathbf{x}) = w_0 + w_1 q_1 + w_2 q_2 + w_3 \dot{q}_1 + w_4 \dot{q}_2 + w_5 \ddot{q}_1 + w_6 \ddot{q}_2$$

where the input is $\mathbf{x} = [q_1 \;\; q_2 \;\; \dot{q}_1 \;\; \dot{q}_2 \;\; \ddot{q}_1 \;\; \ddot{q}_2]^\top$, q_1 is the angle between the first link and horizontal with counterclockwise positive, q_2 is the angle between the second link and first link with counterclockwise positive, the joint velocities are $\dot{q}_1$ and $\dot{q}_2$, the joint accelerations are $\ddot{q}_1$ and $\ddot{q}_2$, $w = [w_0 \;\; w_1 \;\; w_2 \;\; w_3 \;\; w_4 \;\; w_5 \;\; w_6]^\top$ is a vector of parameters of the local model, and $\phi(\mathbf{x}) = [1 \;\; q_1 \;\; q_2 \;\; \dot{q}_1 \;\; \dot{q}_2 \;\; \ddot{q}_1 \;\; \ddot{q}_2]^\top$ is a vector or basis functions of the local model.

(a) Locally weighted regression. Load the file "doublependulumdata.mat" into your Matlab workspace. It contains vectors of joint positions, velocities, and accelerations along with torques sampled from the motion of a double pendulum. This is your training data. Using the training data and locally-weighted regression do leave-one-out cross validation (LOOCV) and compute the RMS error of your predictions. This means for each of the data points in the training data set use all of the other training data to predict the ouput for left out data point. Use the median rather than the mean to compute RMS to diminish the effect of outliers. Remeber to include the error in predicting torques for both degrees of freedom.
Use a diagonally-weighted Euclidean distance function of the form

$$d = \sqrt{(\mathbf{x} - \mathbf{q})^\top M^\top M (\mathbf{x} - \mathbf{q})}$$

where $\mathbf{q}$ is a query and M is a diagonal matrix with diagonal elements $\sqrt{200}$, $\sqrt{200}$, $\sqrt{10}$, $\sqrt{10}$, 1, and 1. Use a Gaussian kernel function of the form

$$k(d) = e^{-d^2/h}$$

where h is a scaling factor to be chosen.

1. Compute LOOCV RMS errors for $h = 100$, $h = 1,000$, and $h = 10,000$. Report these errors on the command line using
   `display(['LOOCV RMS error for h = 100 is',num2str(LOOCV)])`.

2. Using the h that gave the lowest LOOCV RMS error plot your torque predictions against the actual torques from "doublependulumdata.mat". Plot individual circles or stars rather than lines connecting the data points. This should ideally be a straight line with a slope of one and intercept at zero. Title the figure "Part (a2)" and use appropriate axis labels.

(b) Computed Torque Control. Use computed torque control with no feeback to move the double pendulum along the desired trajectory contained in the attached file "desiredtrajecotry.mat". That is, given the desired trajectory use locally-weighted regression to compute the torques needed to move the arm along the desired trajectory. The trajectory is a circle centered at 1 meter above the shoulder of your arm with a radius of 0.25 meters. You will run a simulation of the motion of the pendulum. Use the attached file "doublependulumstep.m" to complete your Euler integration steps along with "compute_accel2.m", which is a helper

function for the Euler integration step. You do not need to attach these files to your submission. Use a simulation time step of 0.001 s and use the initial point in the desired trajectory as the initial condition of your simulation.

1. In one figure plot the torque predictions for both degrees of freedom as functions of time. Also plot lines that are two standard deviations above and below each torque plot. This approximately represents the 95% confidence intervals of your predictions. To do this you will have to compute the variance of the torque prediction at each step in time and take the square root to get the standard deviation. Title the figure "Part (b1)" and include a legend and appropriate axis labels.

2. In another figure plot the actual and desired joint angles as functions of time. Title the figure "Part (b2)" and include a legend and appropriate axis labels.

3. Plot an animation of the pendulum given that the length of the first link is 0.8 meters and the length of the second link is 0.6 meters. Make sure you plot the desired circular path of the end of the pendulum with your animation so it is easy to see if your controller worked. Don't be too concerned if this doesn't work terribly well.

(c) Computed Torque Control with Feedback. Repeat part (b) but add feedback torques τ_{FB} to your open-loop torques τ_{FF} you computed in part (b). Use the following control law

$$\tau = \tau_{FF} + K_p(\mathbf{q_{des}} - \mathbf{q}) + K_d(\dot{\mathbf{q}}_{\mathbf{des}} - \dot{\mathbf{q}})$$

where K_p and K_d are feedback gains to tune.

1. In one figure plot the torque you applied in part (b) and the torque you applied in part (c) for both degrees of freedom as functions of time. If your locally weighted regression worked well, the diffence in torques when adding feedback should not be large. Title the figure "Part (c1)" and include a legend and appropriate axis labels.

2. In another figure plot the actual and desired joint angles as functions of time. Title the figure "Part (c2)" and include a legend and appropriate axis labels.

3. Plot an animation of the pendulum given that the length of the first link is 0.8 meters and the length of the second link is 0.6 meters. Make sure you plot the desired circular path of the end of the pendulum with your animation so it is easy to see if your controller worked.

Problem 2:

You are going to use Gaussian process regression to control a two-link pendulum. You GPR model input is $\mathbf{x} = [q_1 \;\; q_2 \;\; \dot{q}_1 \;\; \dot{q}_2 \;\; \ddot{q}_1 \;\; \ddot{q}_2]^\top$ where q_1 is the angle between the first link and horizontal with counterclockwise positive, q_2 is the angle between the second link and first link with counterclockwise positive, the joint velocities are $\dot{q}_1$ and $\dot{q}_2$, and the joint accelerations are $\ddot{q}_1$ and $\ddot{q}_2$. The output is the vector of joint torques applied to the joints of the pendulum.

(a) Gaussian process regression. Load the file "doublependulumdata.mat" into your Matlab workspace. It contains vectors of joint positions, velocities, and accelerations along with torques sampled from the motion of a double pendulum. This is your training data. Using the training data and Gaussian process regression do leave-one-out cross validation (LOOCV) and compute the RMS error of your predictions. This means for each of the data points in the training data set use all of the other training data to predict the ouput for left out data point. Remeber to include the error in predicting torques for both degrees of freedom. Use the following covariance function.

$$k(\mathbf{x}_*, \mathbf{x}) = p_1 * e^{\frac{(\mathbf{x}-\mathbf{x}_*)^\top (\mathbf{x}-\mathbf{x}_*)}{2*p_2^2}}$$

where $\mathbf{x}$ is a input in the training set, $\mathbf{x}_*$ is a query, p_1 is the magnitude hyperparameter, and p_2 is the length scale. Set the magnitude hyperparameter $p_1 = 20$ and noise in the training data $\sigma^2 = 1$.

1. Compute LOOCV RMS errors for $p_2 = 30$, $p_2 = 50$, and $p_2 = 100$. Report these errors on the command line using
   `display(['LOOCV RMS error for p2 = 30 is',num2str(LOOCV)])`.
2. Using the p_2 that gave the lowest LOOCV RMS error plot your torque predictions against the actual torques from "doublependulumdata.mat". Plot individual circles or stars rather than lines connecting the data points. This should ideally be a straight line with a slope of one and intercept at zero. Title the figure "Part (a2)" and use appropriate axis labels.

(b) Computed Torque Control. Use computed torque control with no feeback to move the double pendulum along the desired trajectory contained in the attached file "desiredtrajectory.mat". That is, given the desired trajectory use Gaussian process regression to compute the torques needed to move the arm along the desired trajectory. Use the p_2 that gave the lowest cross validation error. The trajectory is a circle centered at 1 meter above the shoulder of your arm with a radius of 0.25 meters. You will run a simulation of the motion of the pendulum. Use the attached file "doublependulumstep.m" to complete your Euler integration steps along with "compute_accel2.m", which is a helper function for the Euler integration step. You do not need to attach these files to your submission. Use a simulation time step of 0.001 s and use the initial point in the desired trajectory as the initial condition of your simulation.

1. In one figure plot the torque predictions for both degrees of freedom as functions of time. Also plot lines that are two standard deviations above and below each torque plot. This approximately represents the 95% confidence intervals of your predictions. To do this you will have to compute the variance of the torque prediction at each step in time and take the square root to get the standard deviation. Title the figure "Part (b1)" and include a legend and appropriate axis labels.

2. In another figure plot the actual and desired joint angles as functions of time. Title the figure "Part (b2)" and include a legend and appropriate axis labels.

3. Plot an animation of the pendulum given that the length of the first link is 0.8 meters and the length of the second link is 0.6 meters. Make sure you plot the desired circular path of the end of the pendulum with your animation so it is easy to see if your controller worked. Don't be too concerned if this doesn't work terribly well.

(c) Computed Torque Control with Feedback. Repeat part (b) but add feedback torques τ_{FB} to your open-loop torques τ_{FF} you computed in part (b). Use the following control law

$$\tau = \tau_{FF} + K_p(\mathbf{q_{des}} - \mathbf{q}) + K_d(\dot{\mathbf{q}}_{\mathbf{des}} - \dot{\mathbf{q}})$$

where K_p and K_d are feedback gains to tune.

1. In one figure plot the torque you applied in part (b) and the torque you applied in part (c) for both degrees of freedom as functions of time. If your Gaussian process regression worked well, the diffence in torques when adding feedback should not be large. Title the figure "Part (c1)" and include a legend and appropriate axis labels.

2. In another figure plot the actual and desired joint angles as functions of time. Title the figure "Part (c2)" and include a legend and appropriate axis labels.

3. Plot an animation of the pendulum given that the length of the first link is 0.8 meters and the length of the second link is 0.6 meters. Make sure you plot the desired circular path of the end of the pendulum with your animation so it is easy to see if your controller worked.