Robot Learning

Winter Semester 2017/2018, Homework 1

Prof. Dr. J. Peters, D. Tanneberg, M. Ewerton

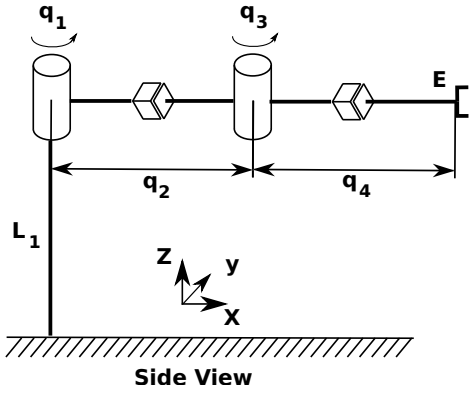
TECHNISCHE UNIVERSITÄT DARMSTADT

Total points: 68 + 10 bonus
Due date: Wednesday, 15 November 2017 (before the lecture)

Name, Surname, ID Number

Problem 1.1 Robotics in a Nutshell [12 Points]

You are considering to buy a new multi-purpose robot platform. Its kinematic chain has two rotational $q_{\{1,3\}}$ and two linear $q_{\{2,4\}}$ degrees of freedom (DoFs), as shown in the figure below. These four joints are actuated with forces and torques of u_i, $i \in \{1,2,3,4\}$. A gripper is mounted on the end of the robot, indicated by the letter **E**. The robot's base is mounted on a table. We assume that the base Cartesian coordinates at the mount are $x_{\text{base}} = [0,0,0]$.



Side View

a) **Forward Kinematics [2 Points]**

Compute the kinematic transformation in the global coordinate system from the base $\boldsymbol{x}_{\text{base}}$ to the end-effector **E**. Write the solution for the $\boldsymbol{x}_{\text{end-eff}} = [x, y, z]^T$ according to the joint values q_i, where $i \in \{1,2,3,4\}$.

b) **Inverse Kinematics [2 Points]**

Define briefly in your own words the inverse kinematics problem in robotics. Can we always accurately model the inverse kinematics of a robot with a function?

c) **Differential Kinematics [4 Points]**

Compute the Jacobian matrix $\boldsymbol{J}(\boldsymbol{q})$ of the robot such that $\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$, where $\dot{\boldsymbol{q}}$ is the first time derivatives of the state vector $\boldsymbol{q}$ of the robot. Explain in a sentence the physical meaning of the Jacobian.

d) **Singularities [3 Points]**

What is the kinematic singularity in robotics? How can you detect it? When does our robotic arm, which was defined above, enter a kinematic singularity?

e) **Workspace [1 Points]**

If your task is to sort items placed on a table, would you buy this robot? Briefly justify your answer.

Problem 1.2 Control [26 Points + 5 Bonus]

In robotic locomotion it is common to abstract from the robot by using inverted pendulum models. In this exercise we will use a planar double inverted pendulum to test different control strategies. Our robot can be controlled by specifying the torque $\mathbf{u} = [u_1, u_2]$ of its motors. Consider that in mechanical systems the torque $\boldsymbol{u}$ is a function of the joint positions $\boldsymbol{q}$, velocities $\dot{\boldsymbol{q}}$ and accelerations $\ddot{\boldsymbol{q}}$, as given by

$$\boldsymbol{u} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}),$$

where $\boldsymbol{M}$ denotes the inertial matrix, $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ the Coriolis and centripetal forces, and $\boldsymbol{g}$ the gravity terms. In the following exercises assume that these terms are given.

For the programming exercises you will use the attached code. We provide skeletons for controlling the system either in joint space (`my_ctl.py`) or in task space (`my_taskSpace_ctl.py`) and a basic functionality for plotting. You can invoke either mode by running `jointCtlComp.py` or `taskCtlComp.py` respectively. Attach a printout with plots and a snippet of your source code for each programming exercise.

a) **PID Controller [2 Points]**

What is the form of a proportional-integral-derivative (PID) controller and how could you use it to control a robot, i.e. what physical quantities could you control? Name one positive and one negative aspect of PID controllers.

b) **Gravity Compensation and Inverse Dynamics Control [4 Points]**

Suppose that you would like to create a control law to set the joint angles on the double inverted pendulum model by controlling the torque of the motors. Write a feedback control law which additionally gravity compensates and then extend it to full inverse dynamics control.

c) **Comparison of Different Control Strategies [12 Points]**

In the following exercise you will investigate the differences of the following control algorithms, P, PID, PD with gravity compensation, and full inverse dynamics. The double pendulum is initiated hanging down, with state $\boldsymbol{q}_{\text{start}} = [-\pi, 0]$. We simulate the system with a time-step $dt = 0.002$ seconds using symplectic Euler integration and run the simulation for $t_{\text{end}} = 3s$.

Implement the control laws by filling the skeleton file `my_ctl.py`. Use the following feedback gains $K_P = 60, K_D = 10, K_I = 0.1$ for the first joint and $K_P = 30, K_D = 6, K_I = 0.1$ for the second one. The target state of the double pendulum is set to $\boldsymbol{q}_{\text{des}} = [-\pi/2, 0]$.

Create (max. 4) plots that compare the different control strategies and analyze the results. It is your choice how to illustrate your results. In your analysis you should include a discussion on the overall performance of each controller. Which controllers manage to go to the desired point, and how does the choice of a controller affects the behavior of the second joint of the pendulum? Additionally discuss which controller you would choose and why. The provided code is able to generate plot but feel free to modify it if you like. Points will be deducted for confusing plots. Do not forget to include your source code in your solutions.

d) **Tracking Trajectories [4 Points]**

Repeat the same experiment but this time use the provided time-varying target trajectory. Create (max 4) plots that compare the different control strategies and analyze the results. In your analysis discuss the overall performance and which controllers track the desired trajectory nicely. Additionally discuss which controller you would choose and why.

e) **Tracking Trajectories — High Gains [4 Points]**

Repeat the same experiment (using the provided trajectory) but this time multiply the gains by ten. Create plots that compare the different control strategies and analyze the results. In your analysis discuss the overall performance and compare it to the previous case. Are there any drawbacks of using high gains?

f) **Task Space Control [5 Bonus Points]**

The robot must now reach a desired position in task space $\boldsymbol{x}_{\text{end}} = [-0.35, 1.5]$. In class we derived the Jacobian transpose, Jacobian pseudo-inverse, and Jacobian pseudo-inverse with damping methods. All of them are implemented in `my_taskSpace_ctl.py`. You are asked to implement also the null-space task prioritization method with a null-space resting posture $\boldsymbol{q} = [0, \pi]$. Run the simulation and plot the initial and final configuration of the robot. Then, change the resting posture to $\boldsymbol{q} = [0, -\pi]$ and redo the plots. Analyze in a couple of sentences your observation. Use the same damping coefficient 10^{-6} and include a code snippet to your solutions.

Problem 1.3 Machine Learning in a Nutshell [30 Points + 5 Bonus]

For this exercise you will use a dataset, divided into training set and validation set (both attached). The first row is the vector $\boldsymbol{x}$ and the second row the vector $\boldsymbol{y}$.
Based on these data, we want to learn a function mapping from $\boldsymbol{x}$ values to $\boldsymbol{y}$ values, of the form $\boldsymbol{y} = \boldsymbol{\theta}^T \boldsymbol{\phi}(\boldsymbol{x})$.

For all questions requiring a plot, you also have to provide a snippet of your code!

You are allowed to use `scipy.spatial.distance.cdist` and `scipy.exp`.

a) **Supervised vs Unsupervised Learning [2 Points]**

Briefly explain the differences between supervised and unsupervised learning. Is the above a supervised or unsupervised learning problem? Why?

b) **Regression vs Classification [2 Points]**

Supervised learning is typically divided into regression and classification tasks. Briefly explain what are the differences between regression and classification.

c) **Linear Least Squares [4 Points]**

Consider the training set above to calculate features $\boldsymbol{\phi}(x)$ of the form $[\sin(2^i x)]_{i=0\ldots n-1}$. Compute the feature values when n is 2, 3 and 9 (i.e., when using 2, 3 and 9 features). Use the linear least squares (LLS) method to predict output values y for input values $x \in \{0, 0.01, 0.02, \ldots, 6\}$ using the different numbers of features. Attach a single plot showing the three resulting predictions when using 2, 3 and 9 features (i.e., having x and y as axes).

d) **Training a Model [2 Points]**

The root mean square error (RMSE) is defined as $\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i^{\text{true}} - y_i^{\text{predicted}}\right)^2}$, where N is the number of data points. Using the LLS algorithm implemented in the previous exercise, train a different model for each of the number of features between 1 and 9, i.e., [1,2,3...,9]. For each of these models compute the corresponding RMSE for the training set. Attach a plot where the x-axis represents the number of features and the y-axis represents the RMSE.

e) **Model Selection [4 Points]**

Using the models trained in the previous exercise, compute the RMSE of each of these models for the validation set.
Compare in one plot the RMSE on the training set and on the validation set. How do they differ? Can you explain what is the reason for these differences? (Hint: remember the plot from Exercise c)) What is the number of features that you should use to achieve a proper modeling?

f) **Cross Validation [8 Points]**

K-fold cross validation is a common approach to estimate the test error when the dataset is small. The idea is to randomly divide the training set into K different datasets. Each of these datasets is then used as validation set for the model trained from the remaining $K-1$ datasets. The resulting vector of errors $\boldsymbol{E} = [e_1 ... e_K]$ can now be used to compute a distribution (typically by fitting a Gaussian distribution). When K is equal to the number of data points, K-fold cross validation takes the name of leave-one-out cross validation (LOO).

Apply LOO using only the training set and compute the mean/variance of the RMSE for the learned models. Repeat for the models with the number of features between 1 and 9, i.e., [1,2,3...,9]

Attach a plot showing the mean/variance (as a distribution) of the RMSE computed using LOO and having on the x-axis the number of features and on the y-axis the RMSE. Which is the optimal number of features now? Discuss the results obtained and compare against model selection using train/validation set.

g) **Kernel Functions [2 Points]**

A kernel function $k\left(\mathrm{x}_i, \mathrm{x}_j\right)$ is given by the inner product of two feature vectors. Write out the kernel function for the previous set of features where $n = 3$.

h) **Kernel Regression [6 Points]**

The kernel function in the previous question required explicit definition of the type and number of features, which is often difficult in practice. Instead, we can use a kernel that defines an inner product in a (possibly infinite dimensional) feature space.
Using the training set and an exponential squared kernel $k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \exp\left(-\frac{1}{\sigma^2}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right)$ with $\sigma = 0.15$, predict output values y for input values $x \in \{0, 0.01, \ldots, 6\}$. Attach a plot of your results.
(Hint: use standard kernel regression: $f(\boldsymbol{x}) = \boldsymbol{k}^\mathsf{T}\boldsymbol{K}^{-1}\boldsymbol{y}$ with $\boldsymbol{K}_{ij} = k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ and $\boldsymbol{k}_i = k(\boldsymbol{x}, \boldsymbol{x}_i)$).

Compute the RMSE on the validation set for the kernel regression model. Compare it with the RMSE of the best LLS model you found.

i) **Derivation [5 Bonus Points]**

Explain the concept of ridge regression and why/when it is used. Derive its final equations presented during the lecture.
(Hint: remind that for normal linear regression the cost function is $J = \frac{1}{2}\sum_{i=1}^{N}(f(\boldsymbol{x}_i) - y_i)^2$)
(Hint 2: use matrix notation)