Machine Learning in Robotics
Gaussian Processes

Prof. Dongheui Lee

Institute of Automatic Control Engineering
Technische Universität München

dhlee@tum.de

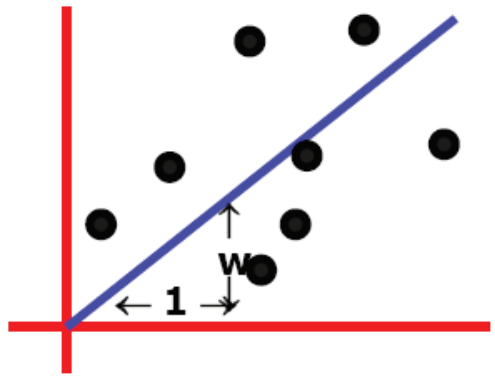
Today's Lecture Outline

- Gaussian Processes: Introduction
- Gaussian Processes for Regression
- Hyper-parameters and their Optimal Selection
- Kernel
- Examples in Robotics

Recall: Linear Regression

Linear regression assumes that expected value of the output given an input is linear.

$$y^{(i)} = wx^{(i)} + \epsilon \tag{1}$$



We just assumed the underlying function $f(x)$ is *linear*. What if the function is cubic, or non-polynomial? Furthermore, if such class model is not known a priori, how can we formulate a hypothesis over the data and maintain general considerations?

Gaussian Processes: definition

Gaussian is widely used to describe **a distribution over vector**:

$$y \sim \mathcal{N}(\mu, \Sigma)$$

Similarly, a Gaussian Process defines **a distribution over functions** $p(f(\boldsymbol{x}))$, where $f(\boldsymbol{x})$ is a scalar function mapping some input space $\mathcal{X}$ to $\mathbb{R}$.

$$f : \mathcal{X} \to \mathbb{R}$$

where $f(\boldsymbol{x})$ is a random variable and that it can be an infinite-dimensional quantity.

$$f \sim \mathcal{GP}(m, K)$$

Definition: *A stochastic process is a Gaussian process if for any finite subset* $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\} \subset \mathcal{X}$, *the marginal distribution* $p(f(\boldsymbol{x}))$ *over that finite subset has a multivariate Gaussian distribution*

Gaussian Processes

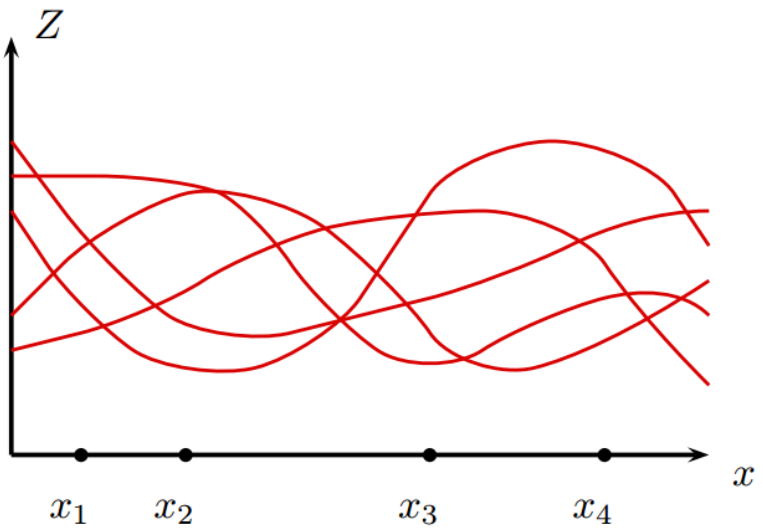
A GP is a Gaussian distribution over functions:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$$

A GP is parameterized by a mean function $m(\boldsymbol{x})$, and a covariance function (kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$

$$m(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$$

$$k(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x}') - m(\boldsymbol{x}'))^T]$$



Gaussian Processes in Practice

A training dataset is given $(\boldsymbol{x}^{(i)}, f(\boldsymbol{x}^{(i)}))$ for $i = 1, ..., n$.

$$f(\boldsymbol{X}) \sim \mathcal{N}(m(\boldsymbol{X}), K(\boldsymbol{X}, \boldsymbol{X}))$$

$$m(\boldsymbol{X}) = (m(\boldsymbol{x}^{(1)}), ..., m(\boldsymbol{x}^{(n)}))^T$$

$$K(\boldsymbol{X}, \boldsymbol{X}) = \begin{bmatrix} k(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(1)}) & \cdots & k(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(n)}) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(1)}) & \cdots & k(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(n)})) \end{bmatrix}$$

Kernel Functions

GP use kernels to lift input data in a higher dimensional feature space (*kernel trick*)

- The kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ is a measure of "closeness" between inputs: Close points $\longrightarrow$ similar predictions
- The kernel function has to generate a symmetric positive semi-definite covariance matrix for any set of points $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\}$

Common Kernel Functions

- Linear: $k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)}$.
- Squared exponential (SE): $k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \sigma_f^2 \exp^{-\frac{1}{2l}\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|^2}$.
- Radial basis function (RBF):

  $k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \sigma_f^2 \exp^{-\frac{1}{2}\sum_{l=1}^{d} w_l \left(x_l^{(i)} - x_l^{(j)}\right)^2}$, where $x_l^{(i)}$ is the l-th component of $\boldsymbol{x}^{(i)}$.

Gaussian Process Regression

We have a training dataset ($\boldsymbol{X}$ and corresponding observations $\boldsymbol{f} = f(\boldsymbol{X})$. Given a test set $\boldsymbol{X}_*$, we want to predict the function output f_*.
The GP defines a joint distribution for $p(\boldsymbol{f}, \boldsymbol{f}_* | \boldsymbol{X}, \boldsymbol{X}_*)$.

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{bmatrix}, \begin{bmatrix} \boldsymbol{K} & \boldsymbol{K}_* \\ \boldsymbol{K}_*^T & \boldsymbol{K}_{**} \end{bmatrix} \right)$$

where $\boldsymbol{\mu} = m(\boldsymbol{X})$, $\boldsymbol{\mu}_* = m(\boldsymbol{X}_*)$, $\boldsymbol{K} = \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$, $\boldsymbol{K}_* = \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*)$, $\boldsymbol{K}_{**} = \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*)$.
To infer $\boldsymbol{f}_*$ (or $p(\boldsymbol{f}_* | \boldsymbol{X}, \boldsymbol{X}_*, \boldsymbol{f})$), the rule for conditional multivariate Gaussians is applied.

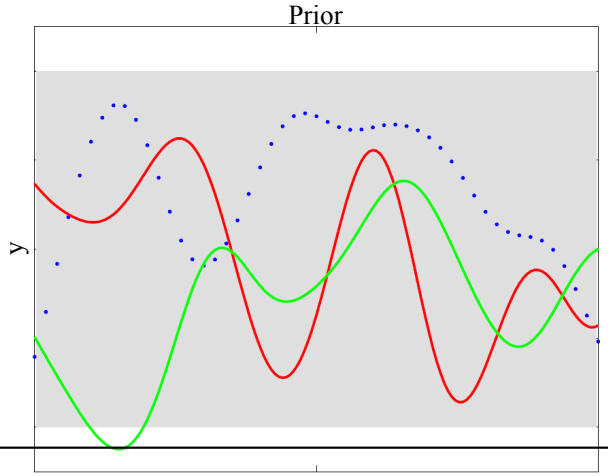
Noiseless GP regression

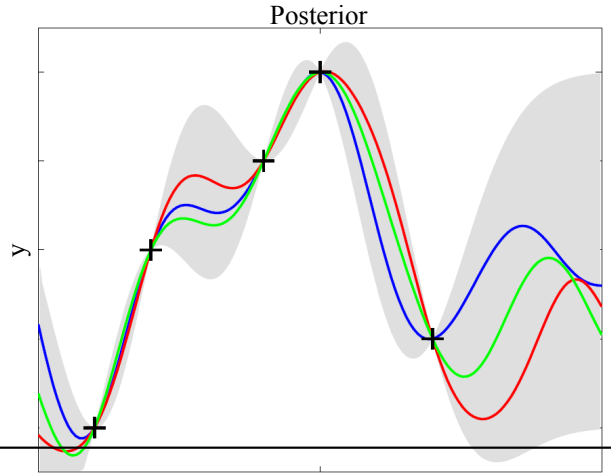
- **Training Data:** $\mathcal{D} = \{\boldsymbol{x}^{(i)}, f^{(i)} = f(\boldsymbol{x}^{(i)})\}_{i=1}^{n}$
- **Prior:**

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{bmatrix}, \begin{bmatrix} \boldsymbol{K} & \boldsymbol{K}_* \\ \boldsymbol{K}_*^T & \boldsymbol{K}_{**} \end{bmatrix} \right)$$

- **Prediction:** Given a test input $\boldsymbol{x}_t$,

$$\begin{aligned} p(\boldsymbol{f}_* | \boldsymbol{X}, \boldsymbol{X}_*, \boldsymbol{f}) &= \mathcal{N}(\boldsymbol{f}_* | \boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*) \\ \boldsymbol{\mu}^* &= \boldsymbol{\mu}_* + \boldsymbol{K}_*^T \boldsymbol{K}^{-1} (\boldsymbol{f} - \boldsymbol{\mu}) \\ \boldsymbol{\Sigma}^* &= \boldsymbol{K}_{**} - \boldsymbol{K}_*^T \boldsymbol{K}^{-1} \boldsymbol{K}_* \end{aligned}$$





Noisy GP regression

- **Training Data:** $\mathcal{D} = \{\boldsymbol{x}^{(i)}, y^{(i)} = f(\boldsymbol{x}^{(i)}) + \epsilon\}_{i=1}^{n} = (\boldsymbol{X}, \boldsymbol{y})$

$$\begin{aligned} f &\sim \mathcal{GP}(0, k(\cdot,\cdot)) \\ \epsilon &\sim \mathcal{N}(0, \sigma_n^2) \end{aligned}$$

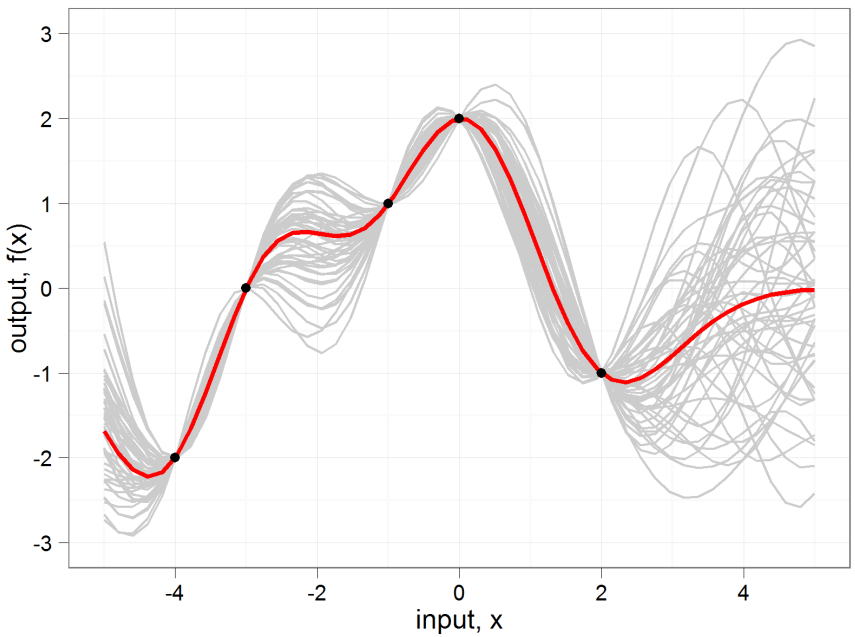
- **Prior:** Given a test input $\boldsymbol{x}_t$

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K} + \sigma_n^2 \boldsymbol{I} & \boldsymbol{K}_* \\ \boldsymbol{K}_*^T & \boldsymbol{K}_{**} \end{bmatrix}\right)$$

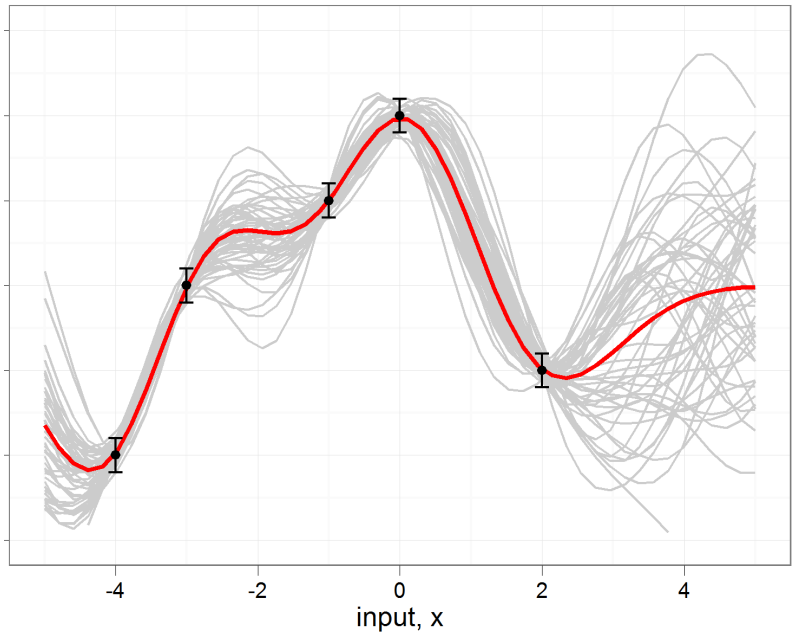
- **Predictive distribution:**

$$\begin{aligned} p(\boldsymbol{f}_* | \boldsymbol{X}, \boldsymbol{X}_*, \boldsymbol{y}) &= \mathcal{N}(\boldsymbol{f}_* | \boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*) \\ \boldsymbol{\mu}^* &= \boldsymbol{\mu}_* + \boldsymbol{K}_*^T \left[\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right]^{-1} (\boldsymbol{y} - \boldsymbol{\mu}) \\ \boldsymbol{\Sigma}^* &= \boldsymbol{K}_{**} - \boldsymbol{K}_*^T \left[\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right]^{-1} \boldsymbol{K}_* \end{aligned}$$

Regression Examples



without noise



with noise

The role of the Hyper-parameters

- Kernels are functions of a set of *hyper-parameters* $\boldsymbol{\theta}$
  - Example: Squared Exponential kernel
    $k(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = \sigma_f^2 \exp^{-\frac{1}{2l}\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|^2}$
    $\boldsymbol{\theta} = [\sigma_f, l]$
- GP regression results strongly depends on the used hyper-parameters

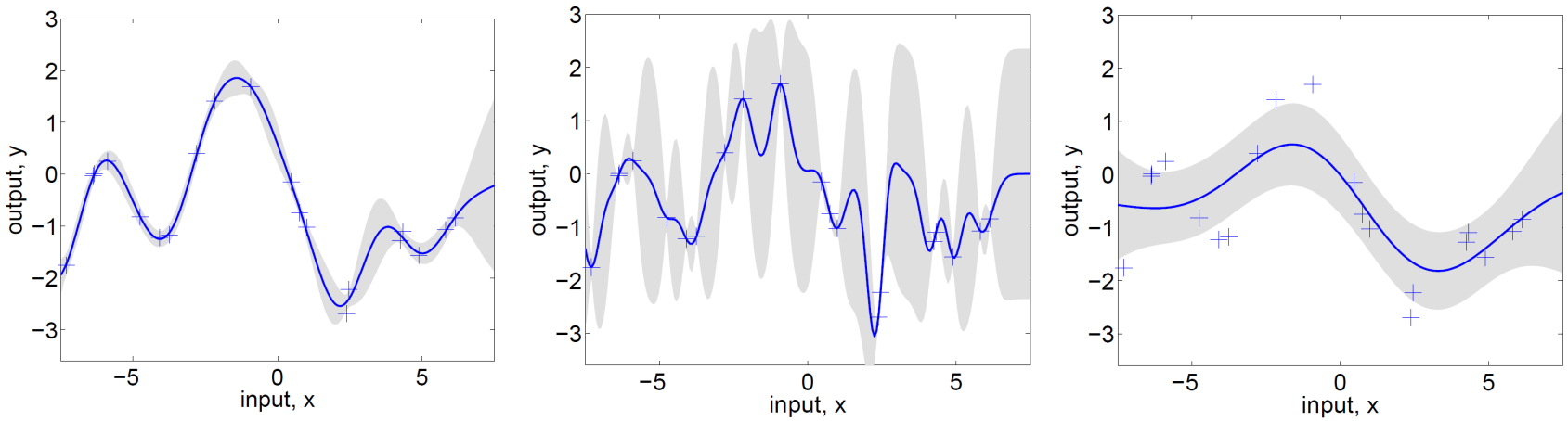


Figure: GP with different hyperparameters. (left) $l = 1$, (middle) $l = 0.3$, (right) $l = 3$

Learning the Hyper-parameters

Hyper-parameters can be learned from observations, which makes GP a parameter-free approach

HOW: Maximize the marginal log-likelihood conditioned on the hyper-parameters $\boldsymbol{\theta}$

1. $p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = \mathcal{N}(0,\boldsymbol{K}_y) = \mathcal{N}(0,\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma_n^2\boldsymbol{I})$
2. $\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^T\boldsymbol{K}_y^{-1}\boldsymbol{y} - \frac{1}{2}\log|\boldsymbol{K}_y| - \frac{n}{2}\log 2\pi$
3. $\frac{\partial}{\partial\theta_j}\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = 0 \longleftarrow$ Gradient descent

The terms of the marginal likelihood have a clear meaning:

- $-\boldsymbol{y}^T\boldsymbol{K}_y^{-1}\boldsymbol{y}/2$ represents the data-fit
- $-\log|\boldsymbol{K}_y|/2$ is the complexity penalty
- $-n\log 2\pi/2$ is a normalization constant

Properties of GPs

- GP is a nonparametric model.
- GPs are universal approximators. (GP can fit arbitrary functions).
- Probabilistic model: it returns measure of uncertainties
- Minimal assumptions over a distribution of functions (i.e., joint Normal)
- GP for regression and classification.
- Kernel functions: Design of kernel functions for application, Hyperparameters can be learned.

Robotics Applications: PILCO

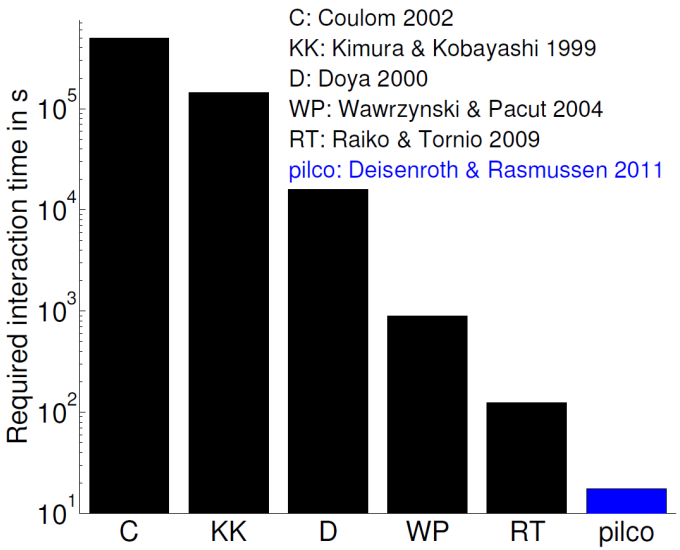
- Model-free reinforcement learning (RL) algorithms usually requires several rollouts to learn a suitable control policy
  - Extremely time consuming on real robots
- PILCO (Probabilistic Inference for Learning Control)
  - Model-based RL
  - GP for long-term predictions and policy evaluation
  - Reduced “real” experience to learn the control policy
- PILCO - Working principle
  1. Perform a rollout (real robot) and learn a model of the robot (GP)
  2. Find an optimal policy for the learned model
  3. Apply the policy to the robot
  4. Finish if task learned, otherwise repeat from 2.

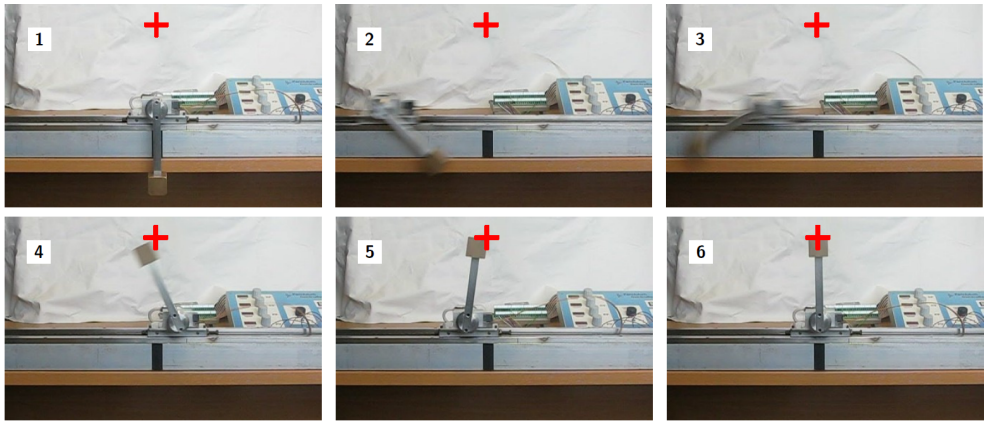
Deisenroth et. al., *Gaussian Processes for Data-Efficient Learning in Robotics and Control*, TPAMI, 2015.

Robotics Applications: PILCO

Task: Swing up a pendulum attached to a cart (cart-pole system, i.e. a cart running on a track and a freely swinging pendulum attached to the cart)

Result: PILCO is able to learn the task significantly faster than state-of-the-art model-based RL approaches





Robotics Applications: LMDS

Locally Modulated Dynamical Systems (LMDS): incrementally refine a set of predefined robotic tasks

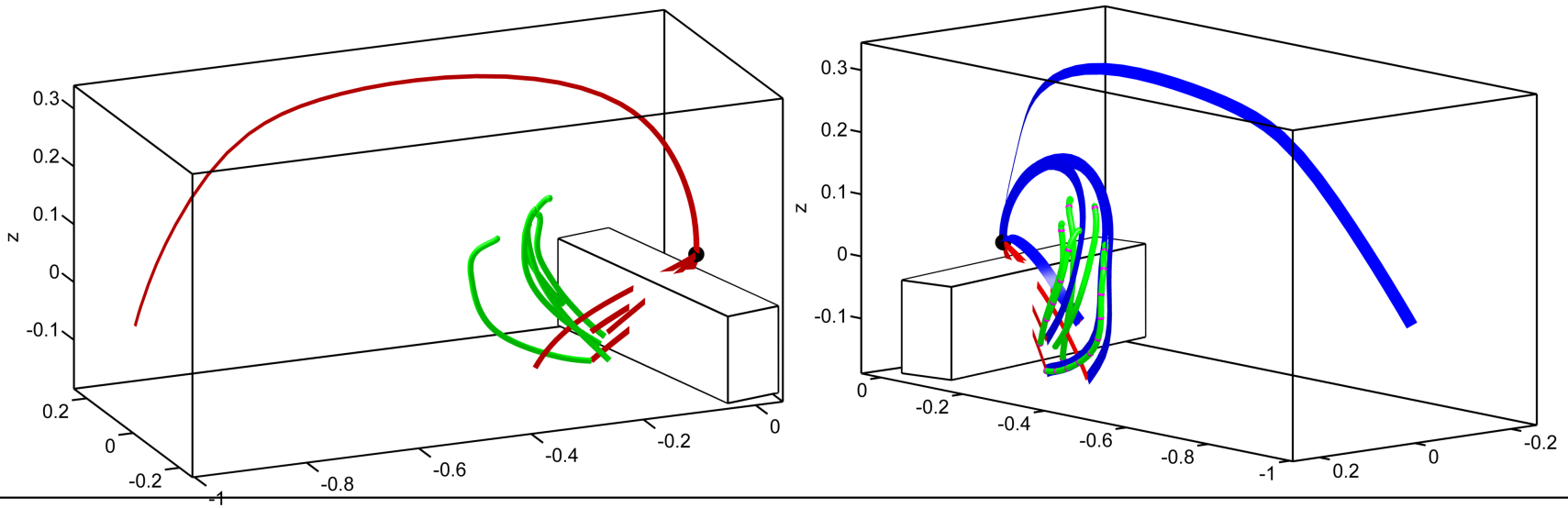
- Predefined tasks are stable dynamical systems $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$, where $\boldsymbol{x}$ is the robot position
- The user can provide kinesthetic demonstrations to modify the task execution
  - Learn (GP) a matrix $\boldsymbol{M}(\boldsymbol{x}) = (1 - \alpha(\boldsymbol{x}))\boldsymbol{R}(\boldsymbol{x})$ from user demonstrations
  - Modify the original task $\dot{\boldsymbol{x}} = \boldsymbol{M}(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x})$ (do not affect the equilibrium points)

Kronander et. al., *Incremental Motion Learning with Locally Modulated Dynamical Systems*, RAS, 2015.

Robotics Applications: LMDS

Task: insert plates into slots in a dish rack.

- The original dynamical achieve the task from different starting positions
- The user provides kinesthetic demonstrations to prevent the robot to hit the rack and the task is correctly executed



Reference

- Rasmussen and Williams, Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning) - Chap. 1, 2, 3