Accelerator Control Using Gaussian Process-Model Predictive Control (GP-MPC) Based Reinforcement Learning (RL)

Su Lei Aye
BE-CSS-DSB
Aug 16th, 2023
Supervised By Michael Schenk, Verena Kain

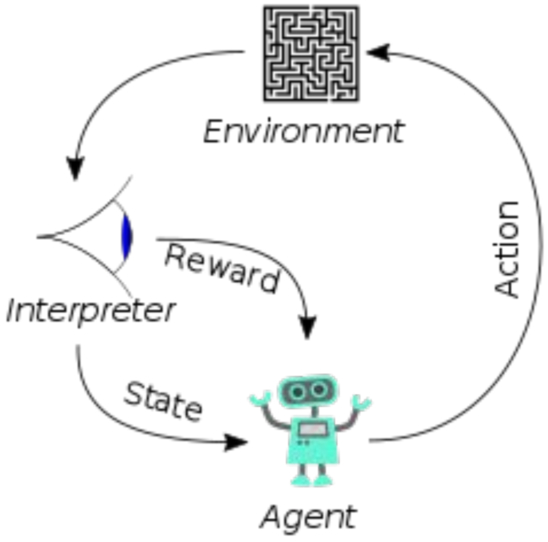
Table of Contents

Introduction

What’s Reinforcement Learning?

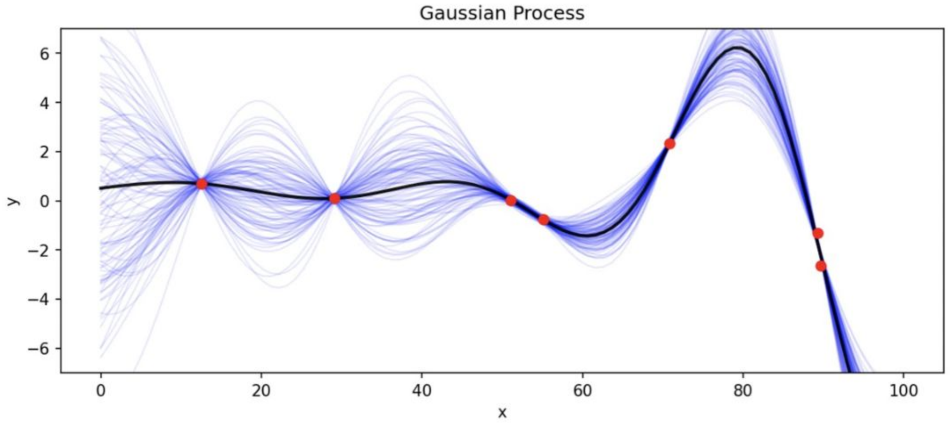
- Machine learning technique
- An agent to learn an interactive environment by trial and error
- Using feedback from its own actions and experiences.

CERN AWAKE Environment



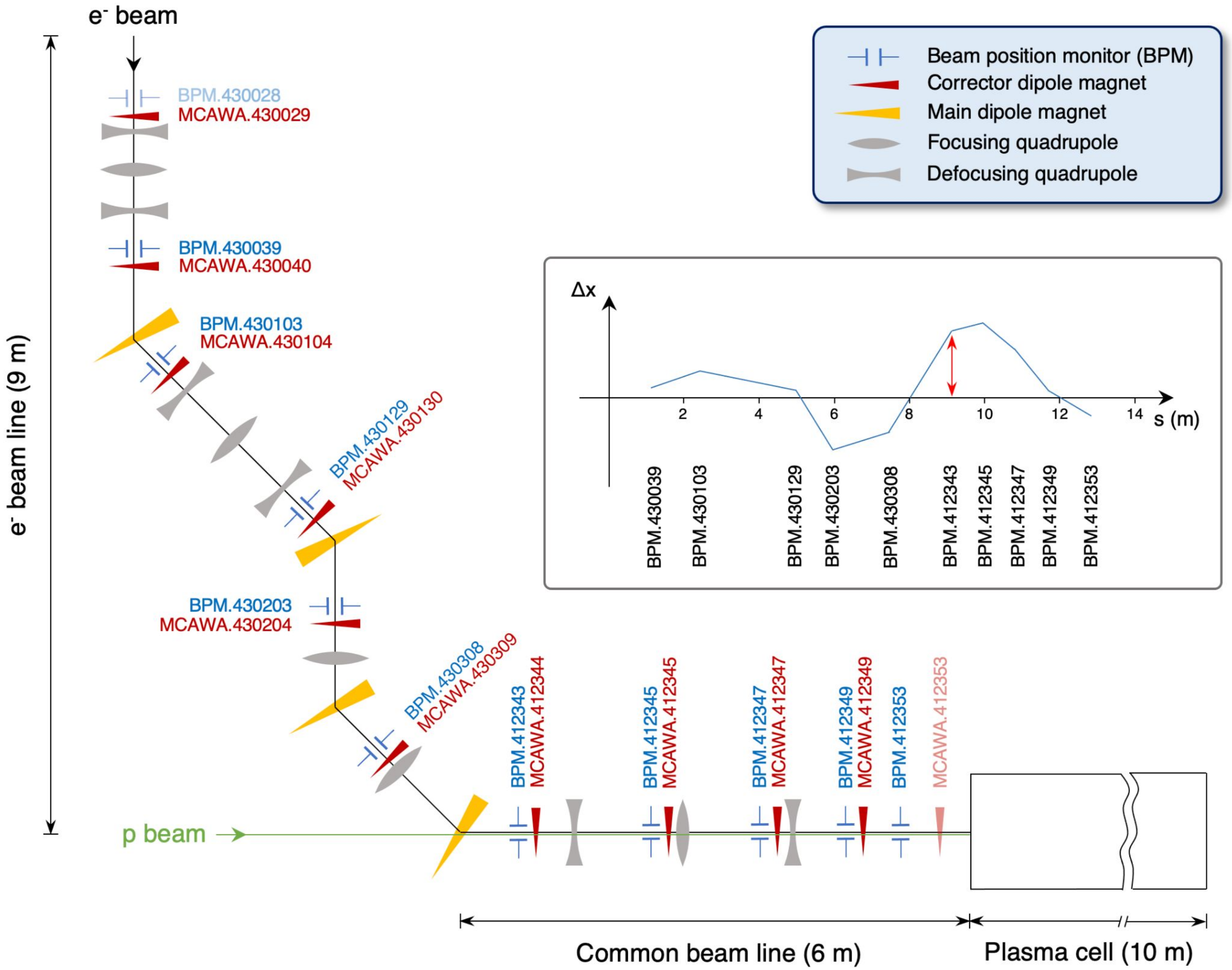
What’s Gaussian Process?

- Probability theory and Statistics Concept
- Represent distribution over a class of functions
- Predict mean values along with uncertainties (confidence intervals)



source: https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292
https://medium.com/geekculture/what-is-gaussian-process-intuitive-explaination-fcee3c78c587

Benchmark: AWAKE electron beam line



PCG (Problems, Challenges & Goal)

Problems: Applying Reinforcement Learning to control accelerator system using Gaussian Process Model-Predictive Control (GP-MPC) algorithm.

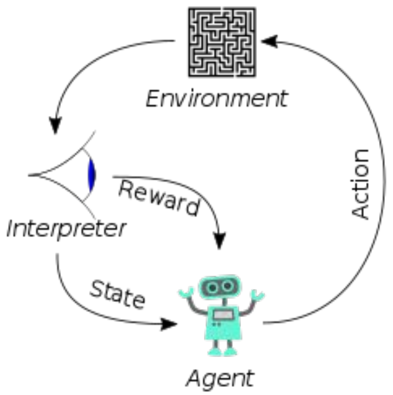
- Use GPs to make models of the real world
  - **1) Transition model:** *(state, action) -> next_state*
  - **2) Reward model:** *next_state -> reward*
- Combine the models to plan ahead and choose the best action with an optimizer (maximize rewards, minimize risks)

Challenges: 1. High sample efficiency (how much training data you need)
2. Include uncertainty and risk estimates
3. Not sufficient beam instrumentation available to define the state.

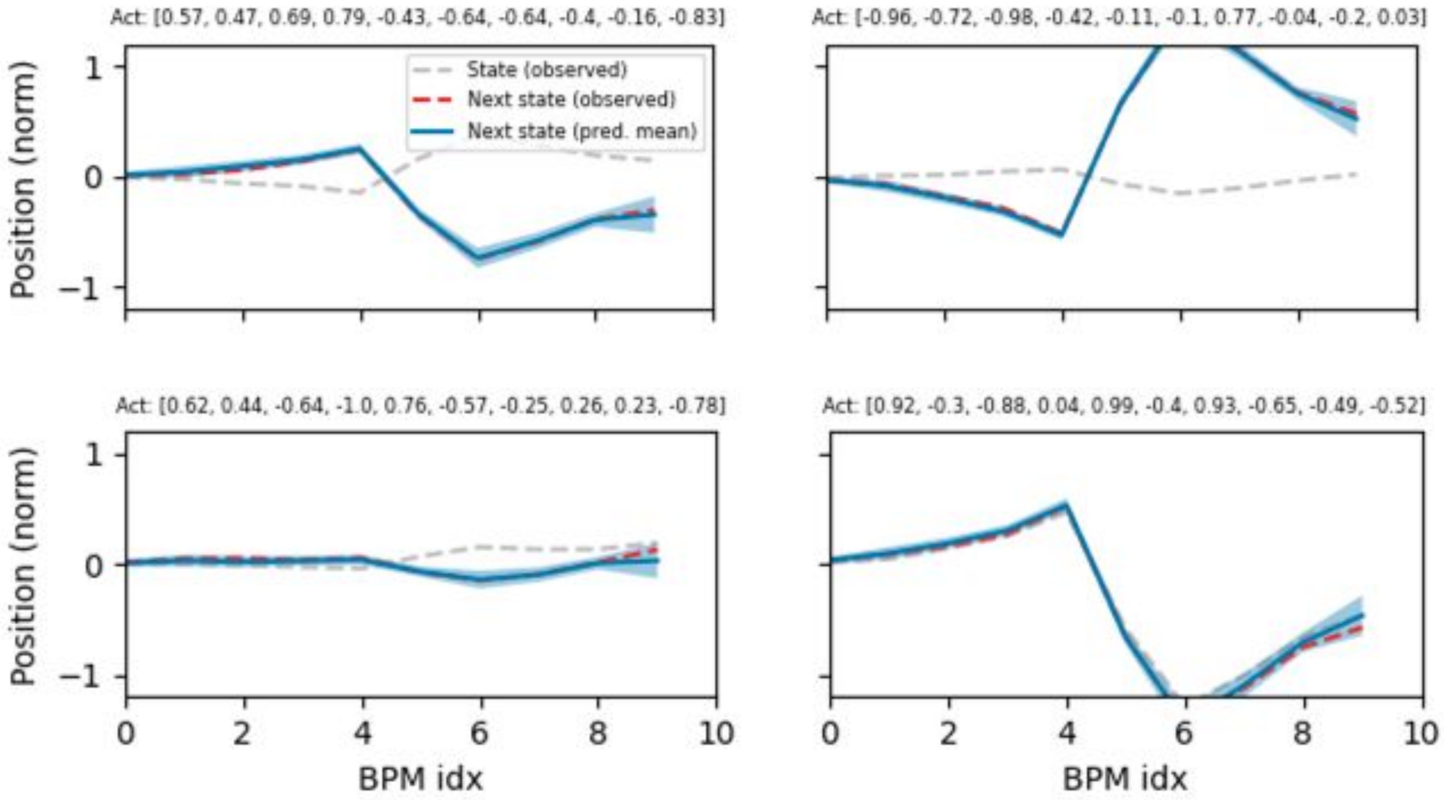
Goal: Implement and test GP-MPC algorithm overcoming the challenges and making it compatible with CERN's Generic Optimisation Framework.

Process

CERN AWAKE Environment



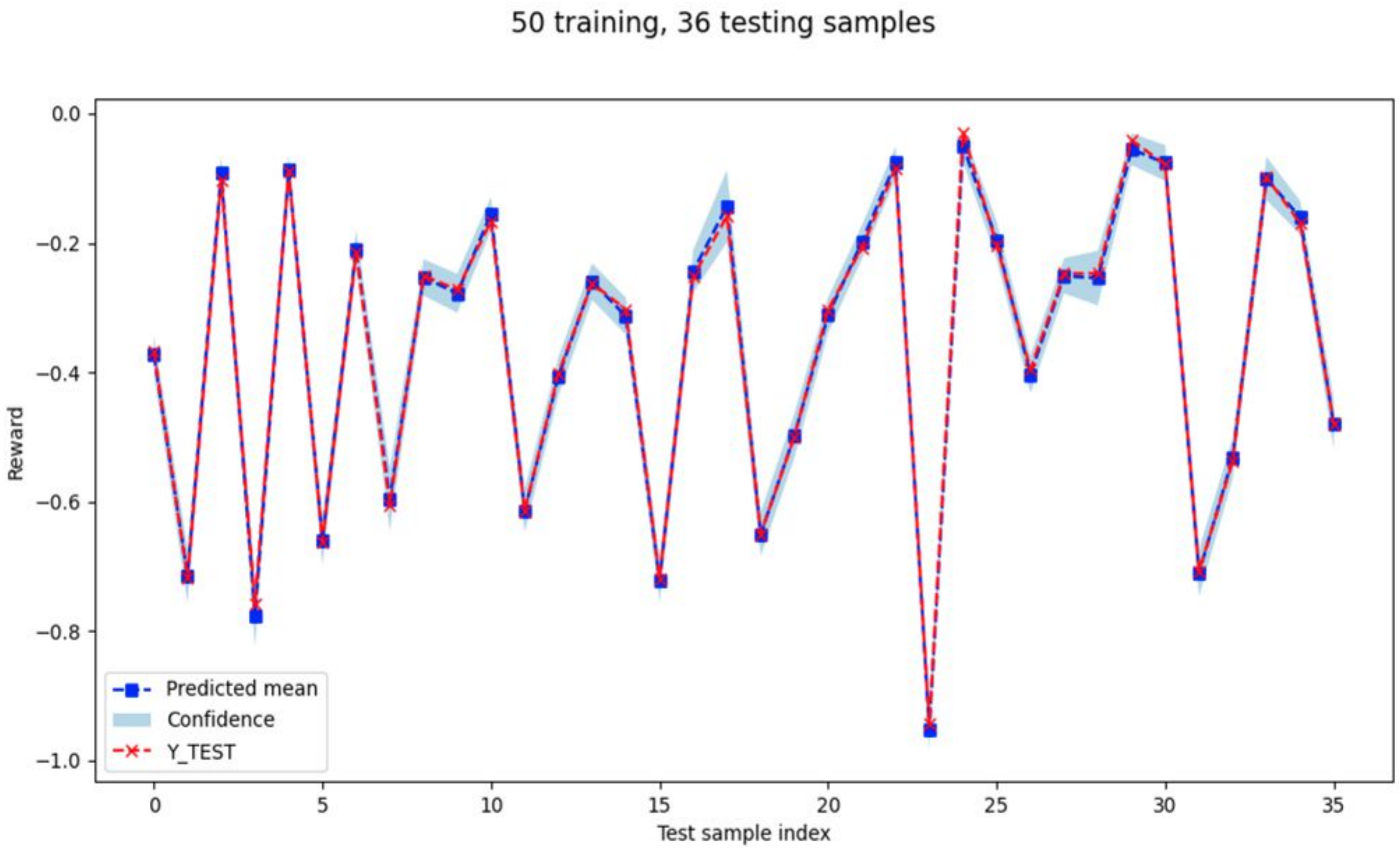
Transition model: *(state, action) -> next_state*



Transition Model: Given state & action (state_action) learn to predict next_states, i.e. how the environment changes when agent takes particular actions.
Purpose: For agent to plan and optimize its action effectively (reward and safety), without applying the action in the real world yet.

CERN openlab

Reward model: *next_state -> reward*



Reward Model: Given the next state learn to predict the Expected Reward.
Purpose: For agent to make informed decision and optimize its behavior to achieve highest rewards.

GP-MPC Based RL

Combined transition & reward models *-> best action (maximize rewards, minimize risks)*

Methodology: acq = OurAcqFunc(self.transition_gp, self.reward_gp, self.beta)

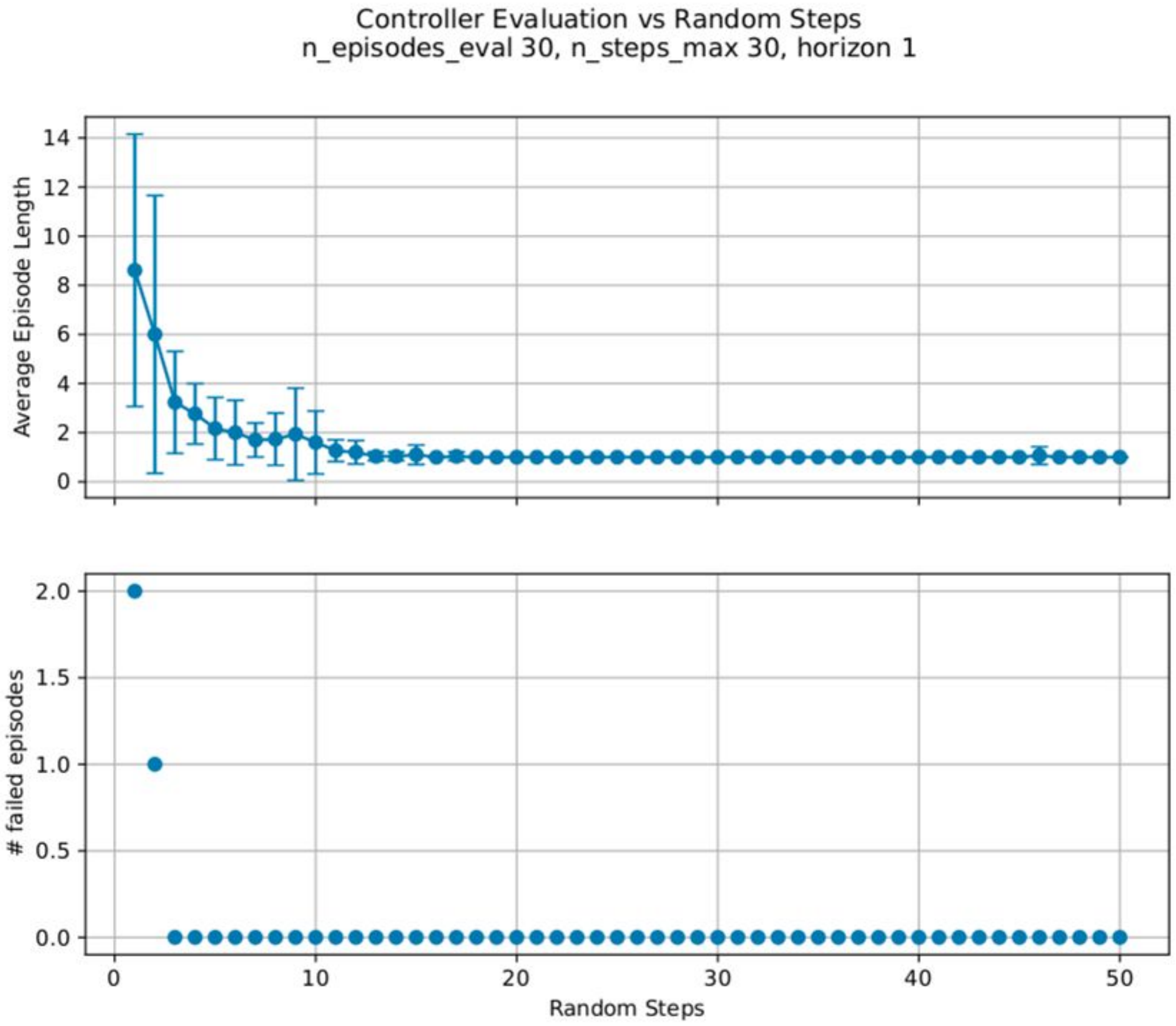
FIRST ITERATION:

```
Initial true reward -0.22491701117210205
Initial pred. reward -0.6606791
ITERATION: 0
  Current state: tensor([-0.0222, -0.0496, -0.0729, -0.1020, -0.1755,  0.2035,  0.4620,  0.3629,
         0.2181,  0.1493])
  Best action: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  Next state: [-0.02216892 -0.04955467 -0.07285278 -0.10200015 -0.17547456  0.20349273
  0.4620307   0.36290238  0.21806641  0.14931128]
  Reward: -0.224917011172102
  Reward (pred): -0.6605854034423828
```

AFTER 18th ITERATION:

```
ITERATION: 18
  Current state: [-0.02216892 -0.04955467 -0.07285278 -0.10200015 -0.17547456  0.20349273
  0.4620307   0.36290238  0.21806641  0.14931128]
  Best action: [-1.43790405e-04  7.53724006e-04 -6.40631093e-04 -6.73291034e-04
  5.24910777e-04 -7.68504165e-05  2.63739254e-04  3.77769772e-04
 -1.37776166e-04 -5.93227664e-05]
  Next state: [-0.02217285 -0.04954173 -0.07284082 -0.10200222 -0.17548277  0.20349988
  0.46205418  0.36293105  0.21810425  0.1493685 ]
  Reward: -0.22493467551064225
  Reward (pred): -0.22495219111442566
```

Result



High Sample Efficiency: only 13 random initial samples needed to have successful controller.

Successful Controller –> agent can solve an episode in one step, i.e. from any beam trajectory it only needs one iteration to reach the objective.

Simulation Variables

random_steps_array = np.arange(1, 51, 1)
n_steps_max = 30
n_episodes_eval = 30
horizon = 1

CERN openlab

Thanks!

QUESTIONS?

saye@cern.ch
suaye@ucsd.edu