Second Milestone Presentation

Moving 2

Technische Universität Berlin

SESE

Structure

1. System components update
2. Reinforcement learning algorithm
3. Schedule
4. Sources

System components update

Refresher: RL for locomotion

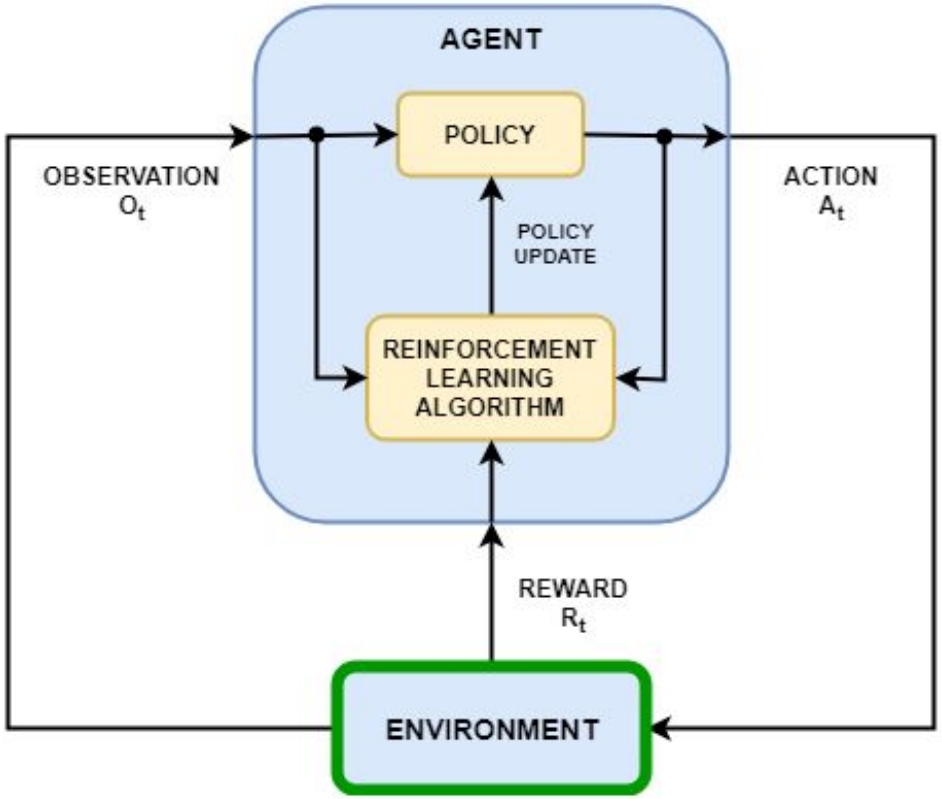
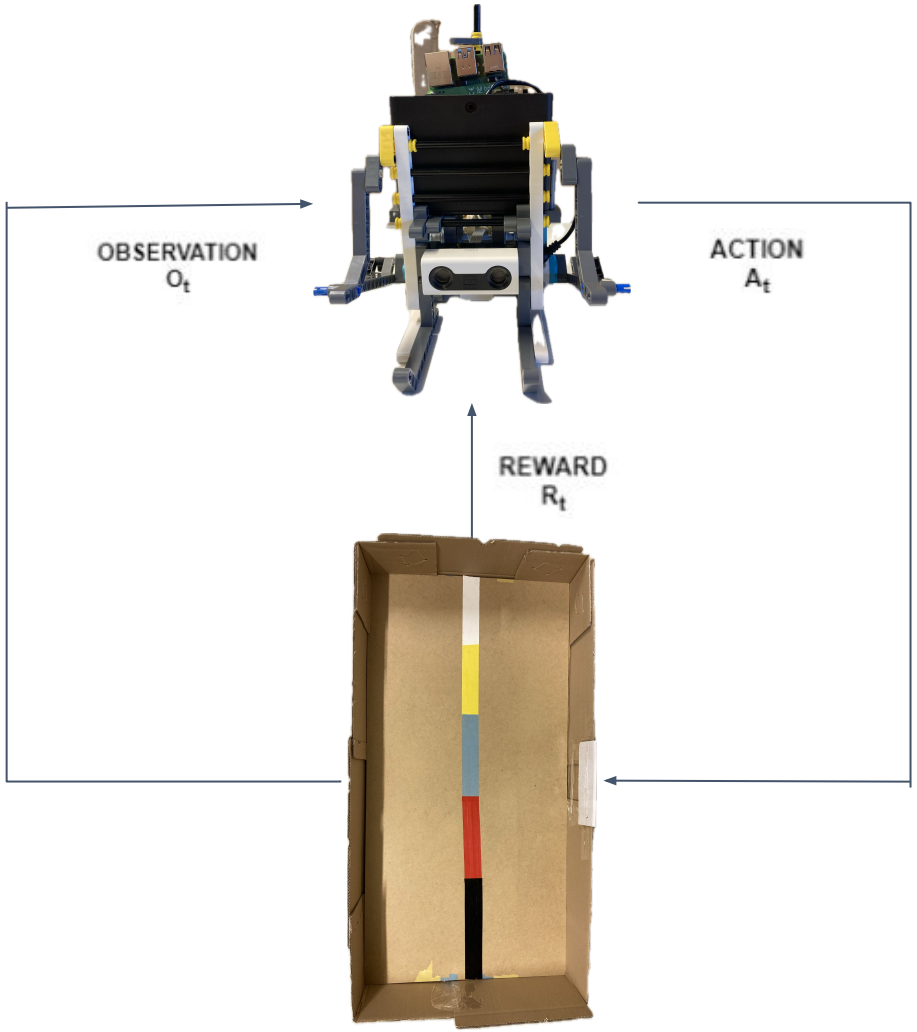
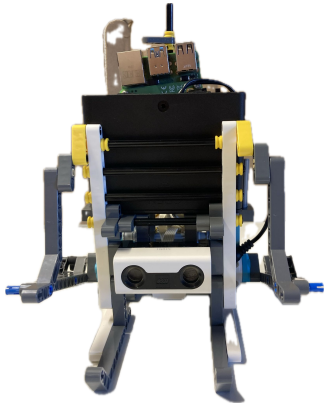


Figure 1: Agent diagram



Agent/ Robot

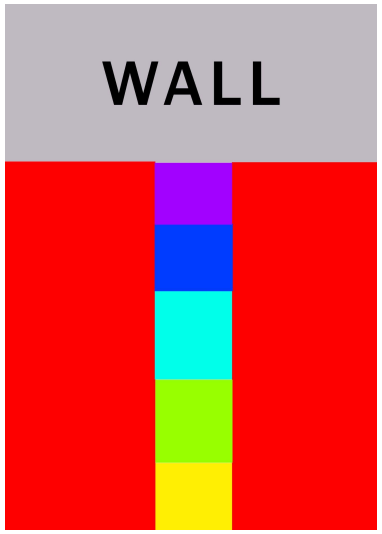
- Sensors
  - Color sensor
  - Distance sensor
- Support for Raspberry Pi
- Support for batteries
- Changed the angles of the two diagonal sticks



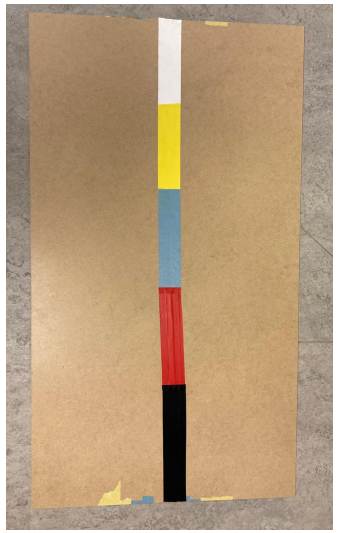
Environment

- Floor: HDF Plate: 1.200 x 600 x 3 mm + Colored Tape
- Walls: Cardboard

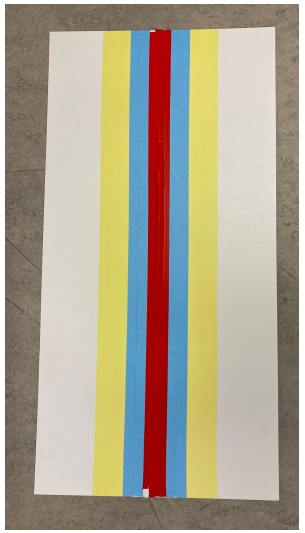
Concept



Option A - Vertical



Option B - Horizontal



Observation space

- **Discrete** observation space **O** = C x D
  - Color sensor: C = {black,red,blue, yellow, white,brown}
  - Distance sensor: D = {dis_0, dis_1..}

distance	Distance in cm
dis_0	[100,80]
dis_1	[80,60]
…	…

color	RGB
red	R:[200,255], G:[0,50], B:[0,50]
blue	R:[0,50], G:[0,50], B:[200,255]
…	…

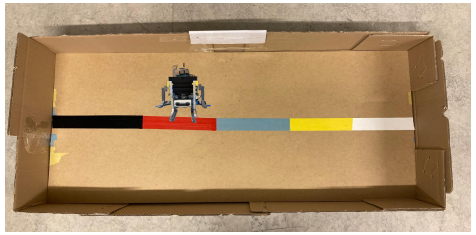
- Observation vector $\mathbf{O_t}$

  O_t = [color,distance]

Example:*

O_{t10}= [red,dis_1]

*Robot started from white



Action space

- **Discrete** action space **A** = speed_r x speed_l

  A = {forward,backward,right,left,stop,undefined}

  *speed ∈ {-1,0,1} with:

  -1: full reverse

  0: stopped

  1: full forward

action(speed*)	speed_r	speed_l
forward	1	1
backward	-1	-1
right	-1	1
left	1	-1
stop	0	0
undefined	0 / -	- / 0

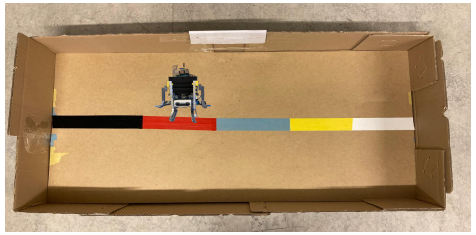
- Action vector A_t

  A_t = [action]

Example:*

A_{t10}= [forward]

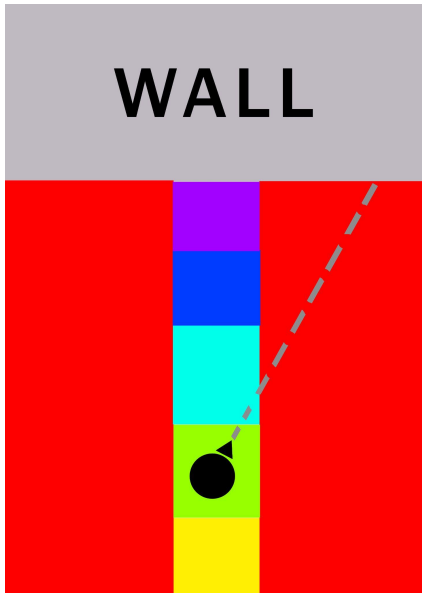
*Robot started from white



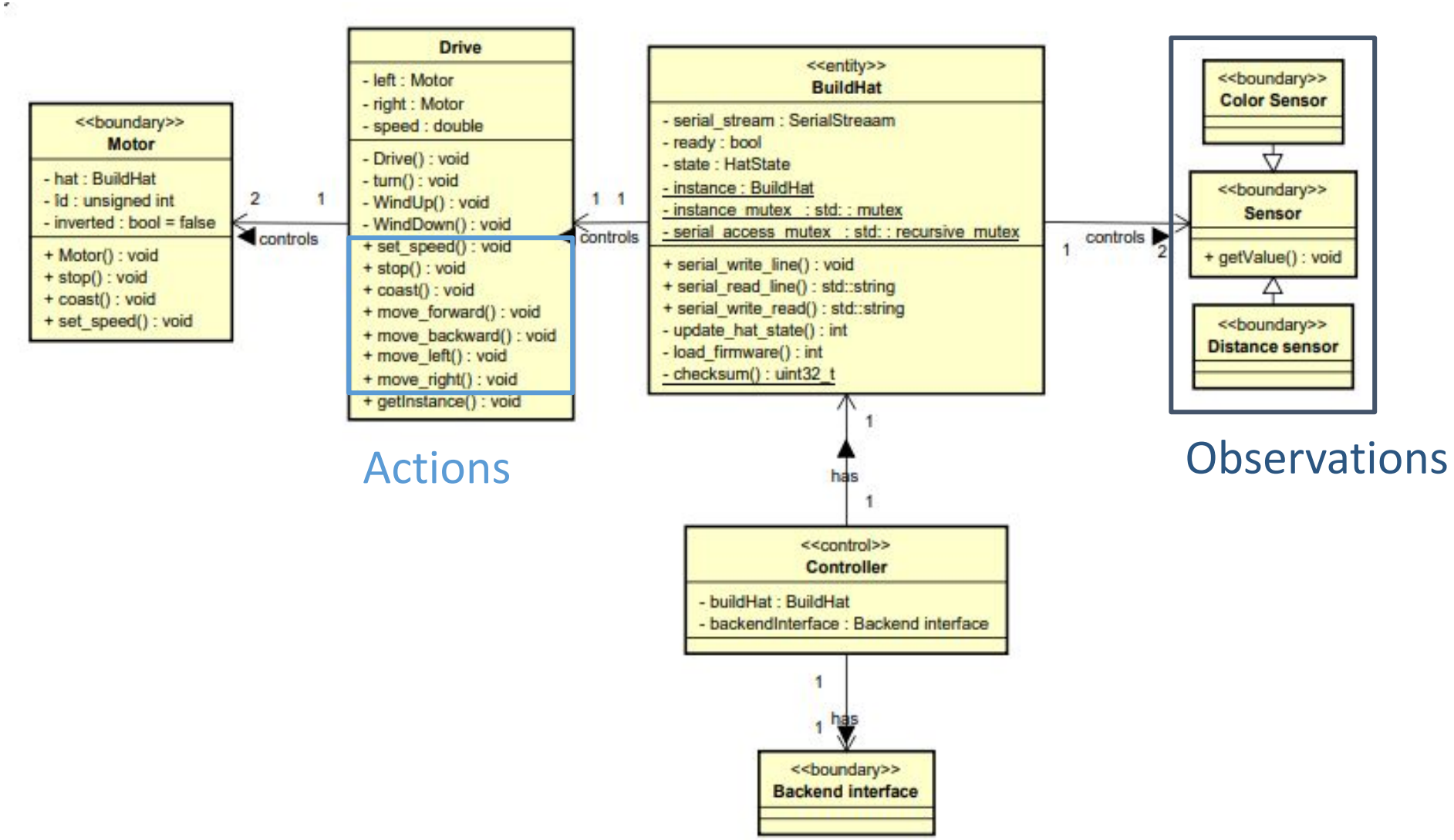
Reward function

$$R = R_{\text{distance}} + R_{\text{color}} + R_{\text{goal}}$$

- $$R_{\text{distance}} = \begin{cases} k_d \times \text{distance} & \text{if distance decreased} \\ -k_d \times \text{distance} & \text{if distance increased} \end{cases}$$
- $$R_{\text{color}} = \begin{cases} k_c & \text{if transitioning to a better color} \\ -k_c & \text{if transitioning to a worse color} \\ 0 & \text{if no transition} \end{cases}$$
- $$R_{\text{goal}} = \begin{cases} k_g & \text{if goal achieved} \\ 0 & \text{otherwise} \end{cases}$$

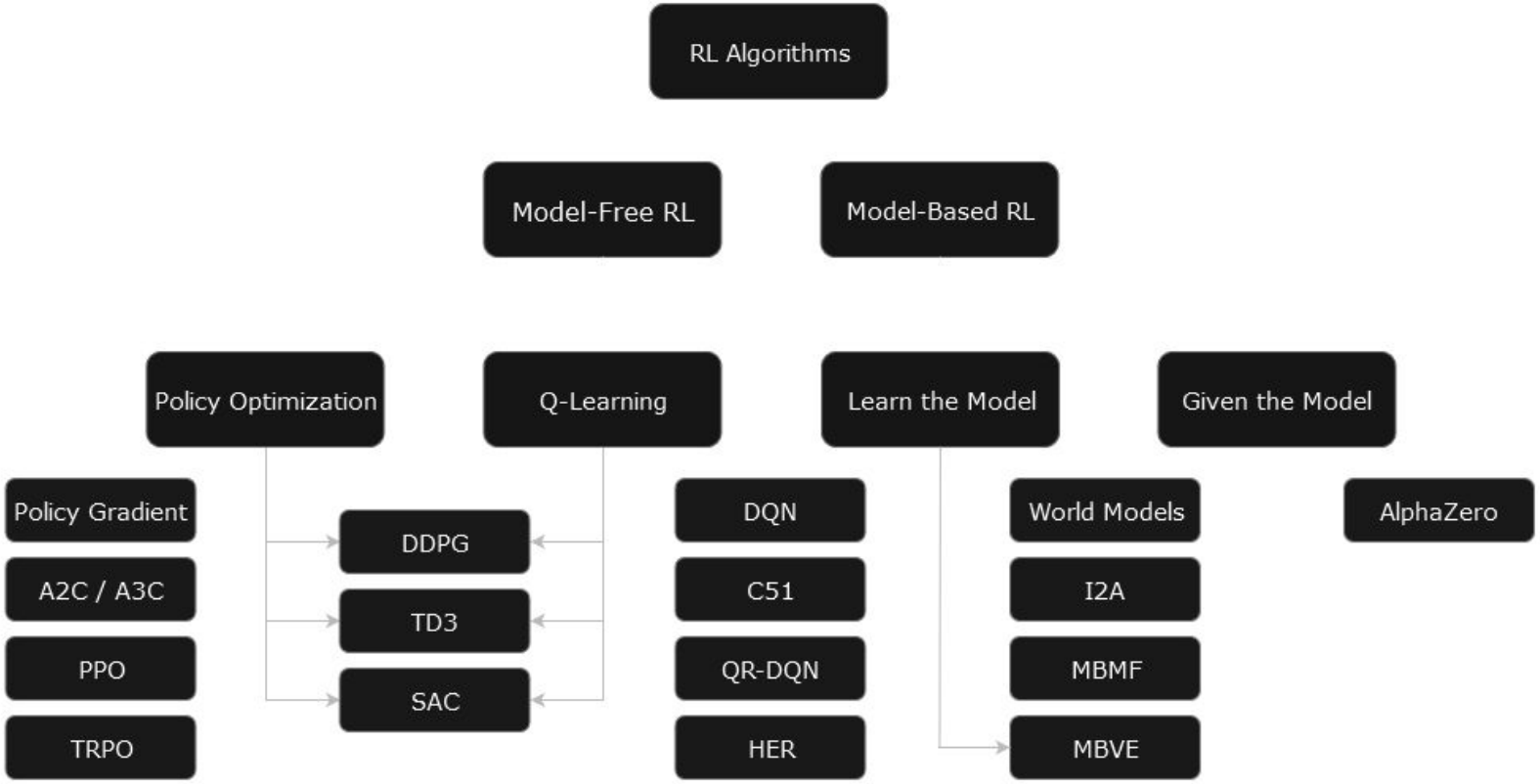


Implementation with buildhat++



Reinforcement Learning Algorithm

Alternatives: PPL, DQN, BCQ, BQ

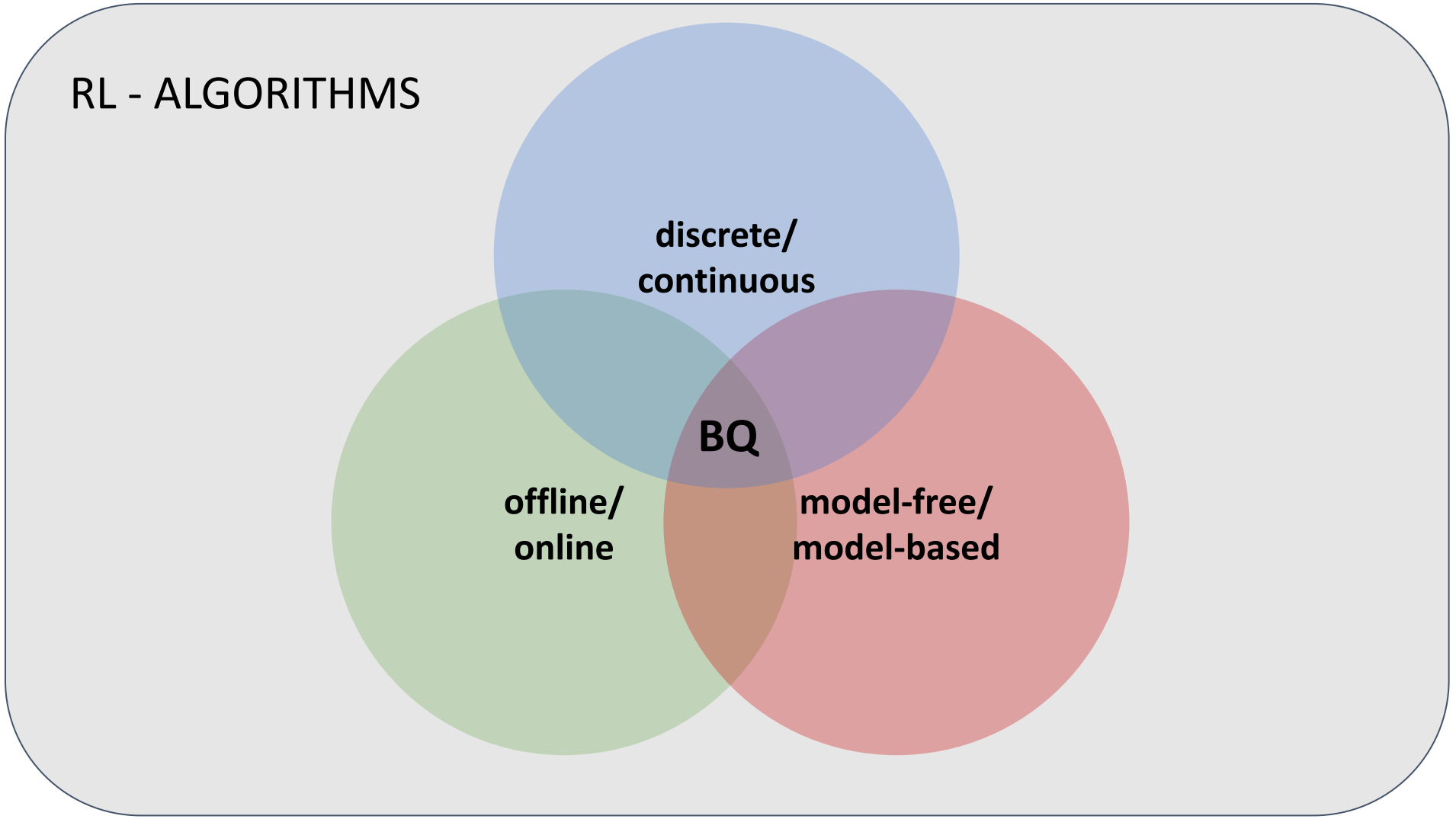


11 : RL Algorithms classification

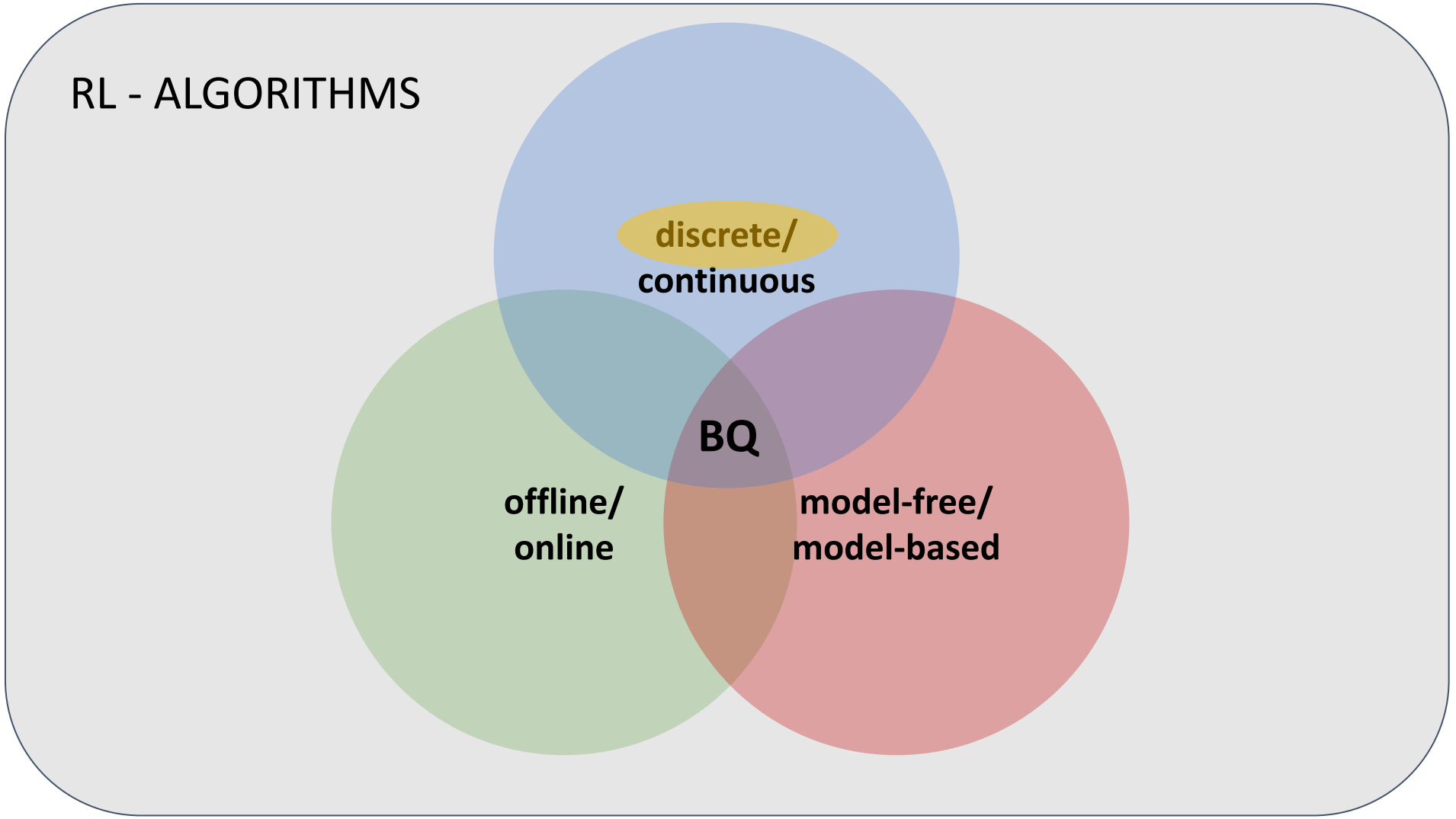
Requirements

RL - ALGORITHMS

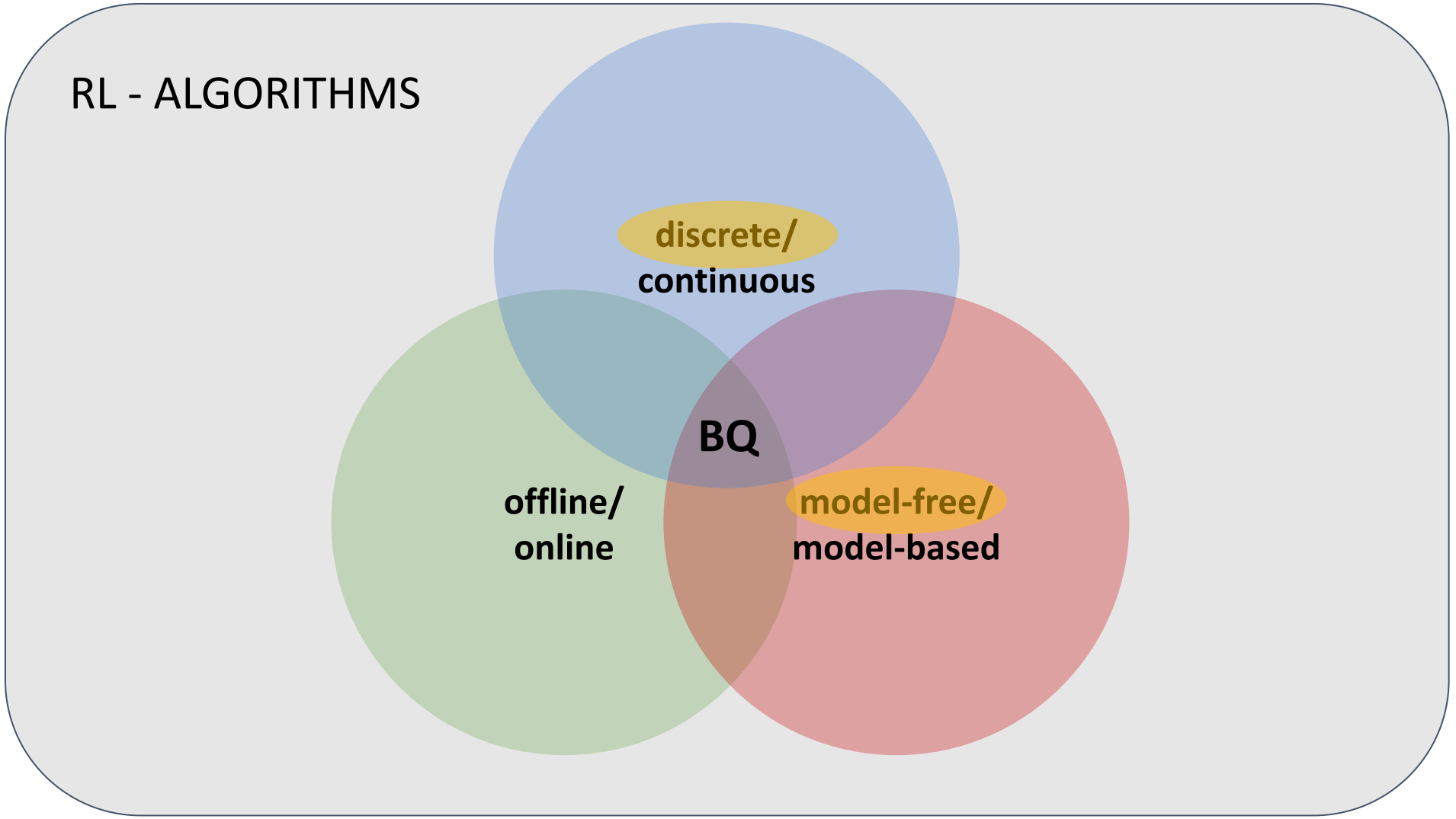
Requirements



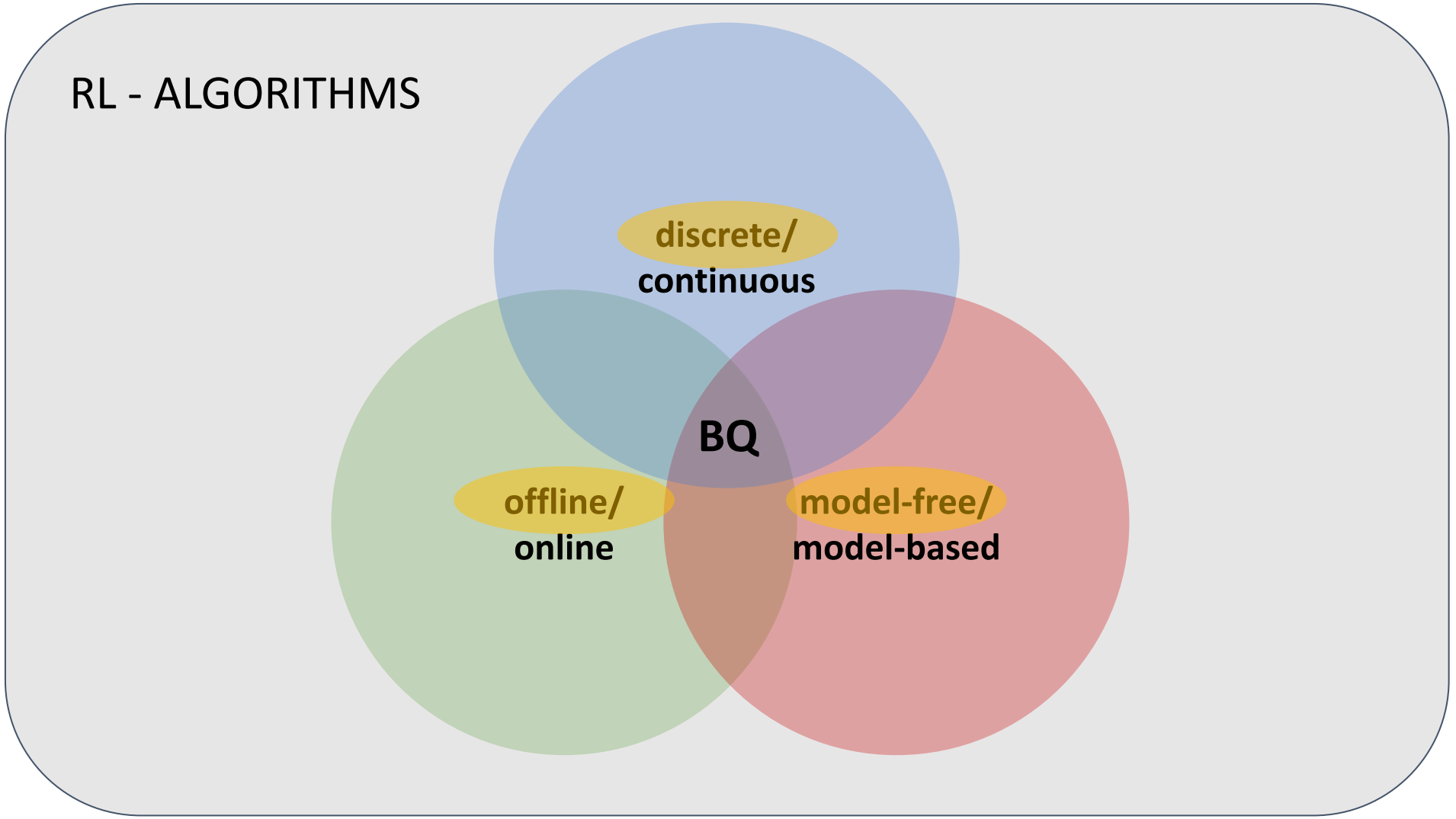
Requirements



Requirements



Requirements



BQ: How it works?

- Q-Learning family
- Commonly used in robotics
- Offline learning solution

Key Features

- Fixed dataset
- Batch updates for Q-values
- Offline learning: reduces risk, improves stability
- Flexible functions

BQ: How it works?

- Two learning stages:
  - Stage One:
    - Independent learning:
      - each learner learns independently
      - Q-Values get evaluated
  - Stage Two:
    - Cooperative learning:
      - learners share Q-Values
      - algorithm balances exploration and exploitation

BQ: How it works? - Challenges

- Bootstrapping problem:
  - extrapolation error
  - overestimation bias

- Solutions:
  - supervise data collection
  - estimate own uncertainty
  - use a stable target network
  - cap Q-Value estimations

Schedule

Milestone 2

	14.05-20.05	21.05-27.05	28.05-03.06
Tasks			
Hardware			
Build an efficient Prototype			
Decide primary System Architecture			
Refine Robot Design			
Software			
Gather broad knowledge about RL Algorithms and how to implement them			
Select RL Algorithm			
Implement RL Algorithm			
Improve RL Algorithm			
Understand C++ Library			
General			
Train Agent			
Define Observation Space (decide which sensors to use and how)			
Define Action Space (Define type of action)			

Milestone 3

	04.06-10.06	11.06-17.06	18.06-24.06	25.06-01.07
Tasks				
Hardware				
Upkeep and last adjustments				
Software				
Implement RL Algorithm				
Improve RL Algorithm				
Refine and test the reward function				
Add more required functions to buildhat library				
Adjust Algorithm as needed				
General				
Train Agent				
Refine and Adapt Environment, Rewards, Spaces and States				

Sources

- Figures
  - Figure 1: Agent diagram
- Papers
  - Abed-alguni, Bilal & Abedalguni, Bilal. (2017). Bat Q-learning Algorithm. Jordanian Journal of Computers and Information Technology. 3. 51. 10.5455/jjcit.71-1480540385.
  - Lange, S., Gabel, T., Riedmiller, M. (2012). Batch Reinforcement Learning. In: Wiering, M., van Otterlo, M. (eds) Reinforcement Learning. Adaptation, Learning, and Optimization, vol 12. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-27645-3_2

Thank you for your attention