DEPARTAMENTO DE INFORMÁTICA PUC-RIO

Reinforcement Learning

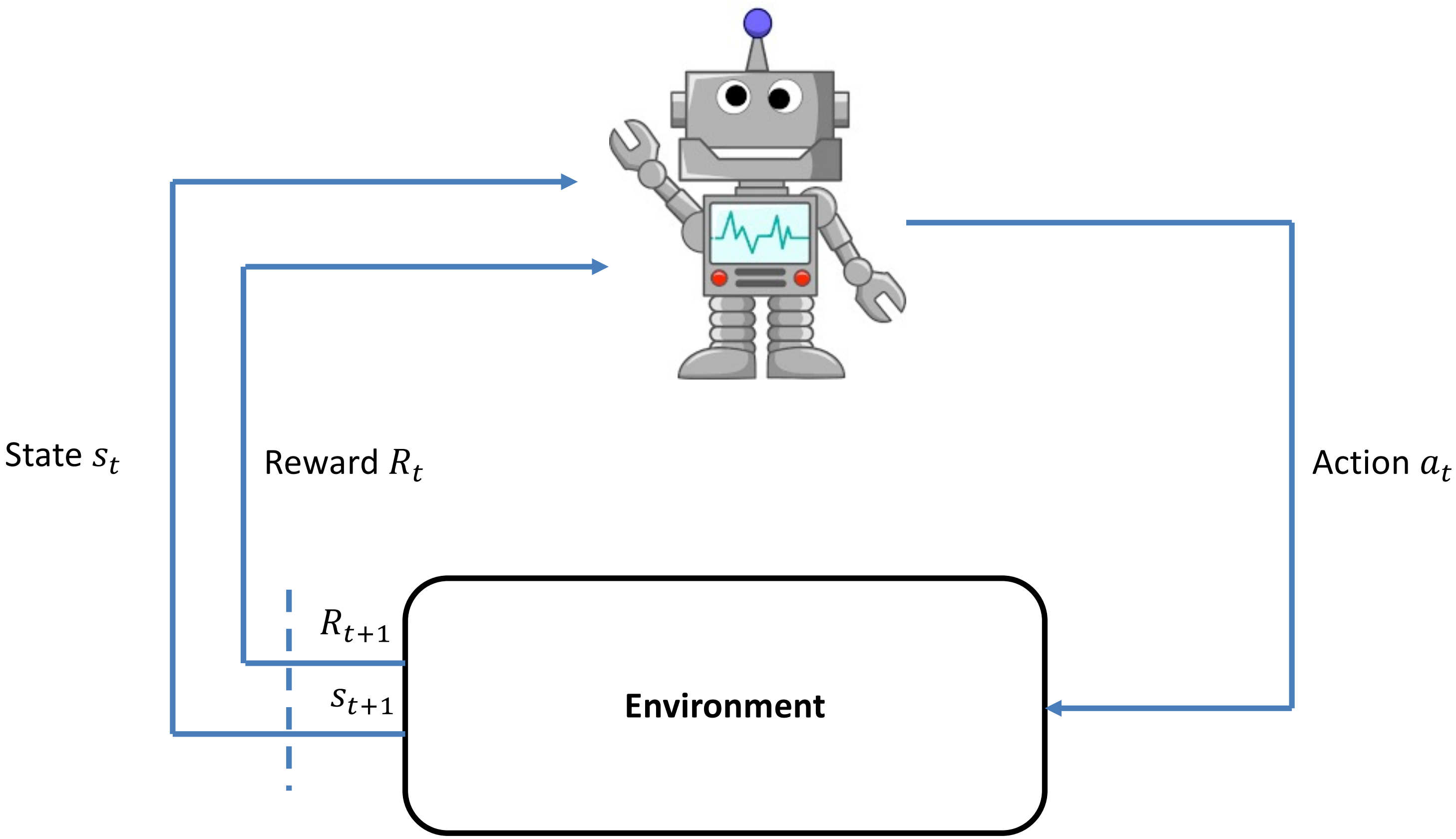
Deep Reinforcement Learning

Alberto Sardinha
sardinha@inf.puc-rio.br

Outline

- **Markov decision problems**
- Reinforcement learning
- Model-based RL

Markov Decision Problems (MDP)



Markov Decision Problems (MDP)

- We can formally define an MDP with following elements:
  - **Discrete time** $t = 0, 1, 2, \ldots$
  - A **discrete set of states** $s \in S$
  - A **discrete set of actions** $a \in A$
  - A **stochastic transition model** $P(s'|s, a)$
    - the world transitions stochastically to state s' when the agent takes action a at state s
  - A **reward function** $R: S \times A \rightarrow \mathbb{R}$
    - An agent receives a reward $R(s, a)$ when it takes action a at state s

Markov Decision Problems (MDP)

- These policies π^* share the same state-value function, called **optimal state-value function**, with the following definition:

$$V^*(s) = \max_{\pi} V^{\pi}(s), \text{ for all } s \in S$$

- These policies π^* also share the same action-value function, called **optimal action-value function**, with the following definition:

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a), \text{ for all } s \in S \text{ and } a \in A$$

Markov Decision Problems (MDP)

- The **optimal state-value function** can be written in a special form without reference to a specific policy (so-called **Bellman equation**):

$$V^*(s) = \max_{a \in A} \left[\boxed{R(s,a)} + \gamma \sum_{s' \in S} \boxed{P(s'|s,a)} V^*(s') \right]$$

- A similar recursive equation holds for the **Q-values**:

$$Q^*(s,a) = \boxed{R(s,a)} + \gamma \sum_{s' \in S} \boxed{P(s'|s,a)} \max_{a' \in A} Q^*(s',a')$$

We need this to compute the optimal policy

Markov Decision Problems (MDP)

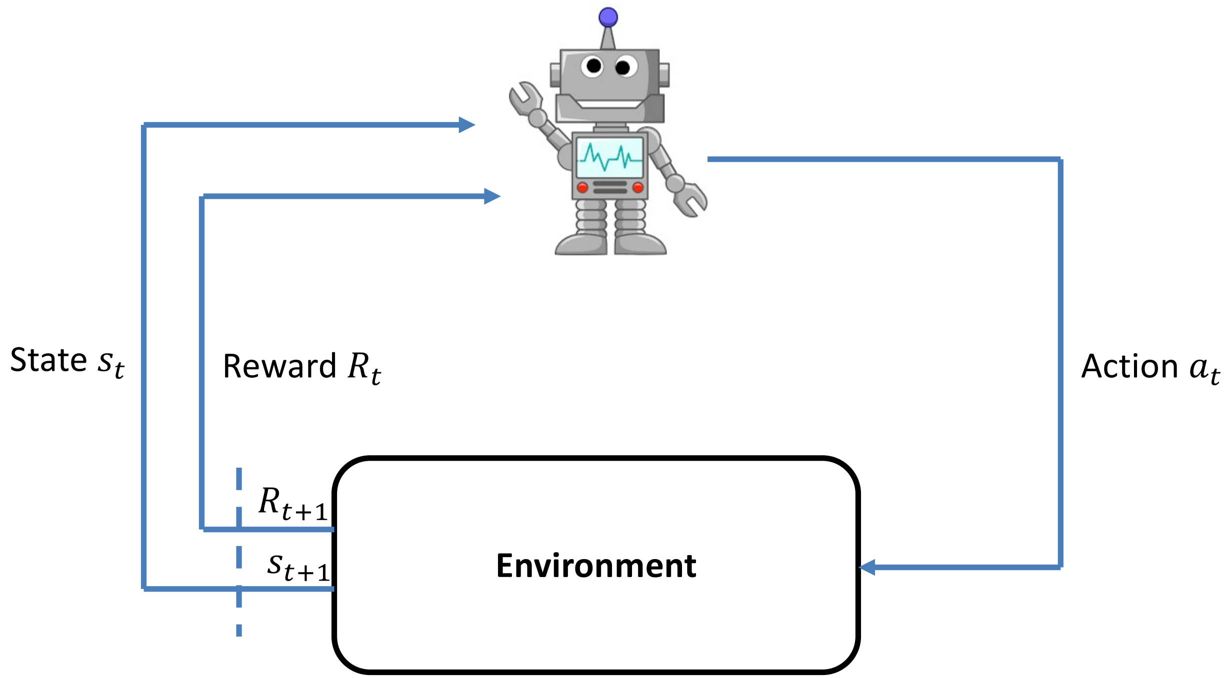
- What happens if we **do not know** the **reward function** and **stochastic transition model**?

Outline

- Markov decision problems
- **Reinforcement learning**
- Model-based RL

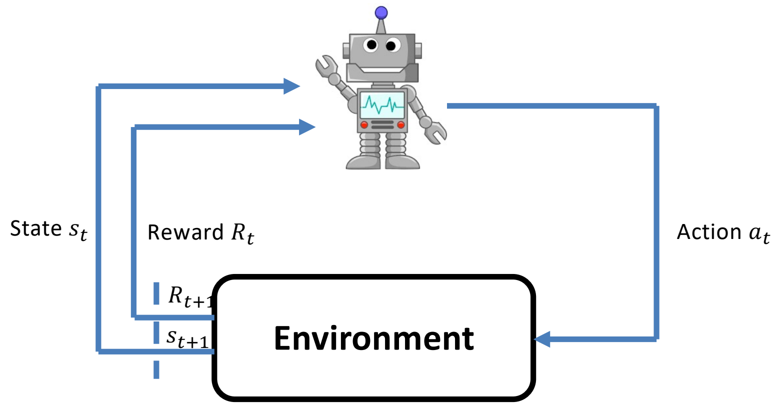
Reinforcement Learning

- Why not **interact with the environment**?



Reinforcement Learning

- Why not **interact with the environment**?
  - At each time step t
    - The agent observes the state s_t
    - The agent takes action a_t
    - The agent observes a reward R_t and the new state s_{t+1}



Reinforcement Learning

- Thus, my data point at each iteration is:

$$(s_t, a_t, R_t, s_{t+1})$$

- And what is the agent's goal?
  - Compute the **optimal policy** with the data points

Reinforcement Learning

- And what is the **difference** between **reinforcement learning** and **supervised learning**?
  - **Supervised learning**
    - Labelled data set (x, y)
    - Learns $\hat{y} = f(x)$
  - **Reinforcement learning**
    - Agent's actions affect the data she will receive
    - Data from experience (interaction with the environment)

      $$(s_t, a_t, \boldsymbol{R_t}, s_{t+1})$$

    - Learns from "reward and punishment"

Reinforcement Learning

- Is this a useful approach? YES!

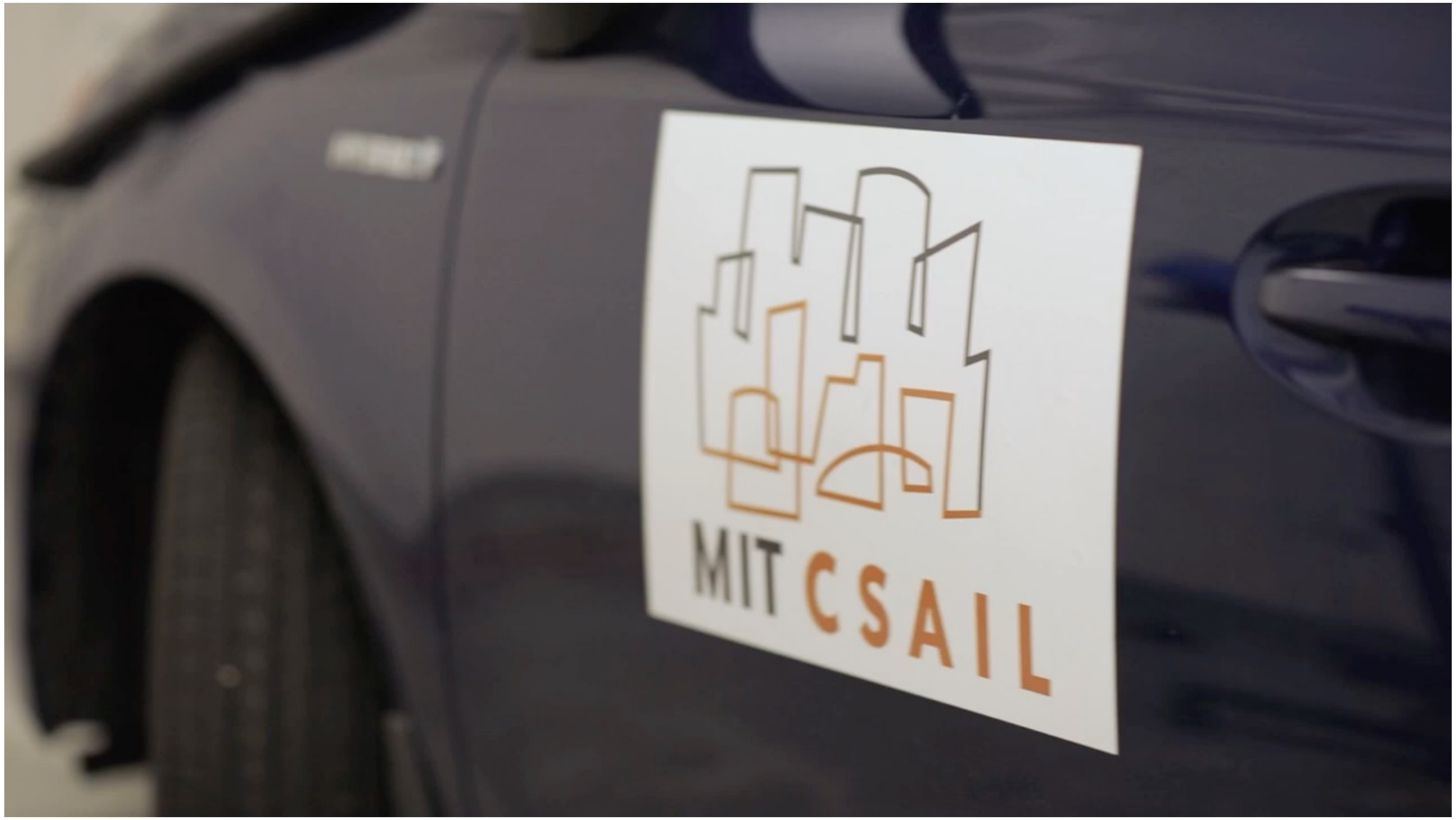
AlphaGo

https://www.youtube.com/watch?v=8tq1C8spV_g

Reinforcement Learning

- Is this a useful approach? YES!

Self-driving cars



https://www.youtube.com/watch?v=pUhckFVXN2A

Reinforcement Learning

- **Model-based RL**
  - Learn de reward function and transition probabilities
  - Use planning (e.g., Value Iteration)
  - Extract policy

- **Value-based RL**
  - Learn the value function directly (Q^*)
  - Extract policy

- **Policy-based RL**
  - Learn the policy directly

Outline

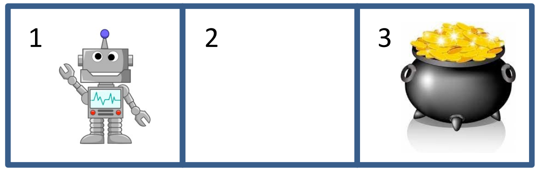
- Markov decision problems
- Reinforcement learning
- **Model-based RL**

Model-based learning

- Basic idea:
  - Run around the environment and collect data points
    - i.e., (s_t, a_t, R_t, s_{t+1})
  - Generate statistics to build a model
    - i.e., $P(s'|s, a)$ and $R(s, a)$
  - Use the model to plan
    - i.e., Value Iteration

Model-based learning

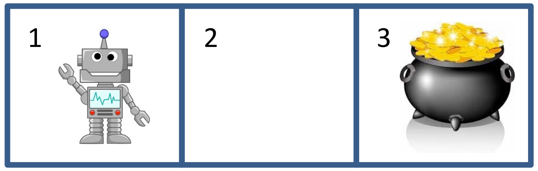
- We have the following MDP:
  - $S = \{1, 2, 3\}$
  - $A = \{left, right\}$
  - $\boldsymbol{P(s'|s, a = left)} = ?$
  - $\boldsymbol{P(s'|s, a = right)} = ?$
  - $\boldsymbol{R(s, a)} = ?$
  - $\gamma = 0.9$



Model-based learning

- Run around the environment and collect data points:

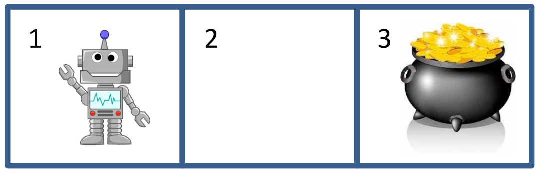
  - $(1, left, 0, 1)$
  - $(1, right, 0, 2)$
  - $(2, left, 0, 2)$
  - $(2, left, 0, 1)$
  - $(1, left, 0, 1)$
  - $(1, right, 0, 1)$
  - $(1, right, 0, 2)$
  - $(2, left, 0, 2)$
  - $(2, right, 0, 3)$
  - $(3, right, 1, 3)$
  - $(3, left, 1, 2)$



Do we need more data points?

Model-based learning

- **Generate statistics to build a model:**
  - **(1, $left$, 0, 1)**
  - (1, $right$, 0, 2)
  - **(2, $left$, 0, 2)**
  - **(2, $left$, 0, 1)**
  - **(1, $left$, 0, 1)**
  - (1, $right$, 0, 1)
  - (1, $right$, 0, 2)
  - **(2, $left$, 0, 2)**
  - (2, $right$, 0, 3)
  - (3, $right$, 1, 3)
  - **(3, $left$, 1, 2)**

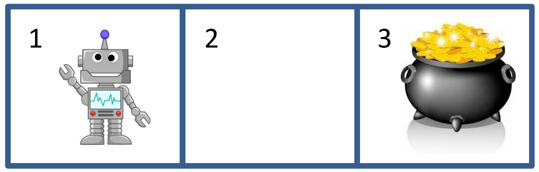


$$P(s'|s, a = left) = \begin{bmatrix} 2/2 & 0 & 0 \\ 1/3 & 2/3 & 0 \\ 0 & 1/1 & 0 \end{bmatrix}$$

Model-based learning

- **Generate statistics to build a model:**

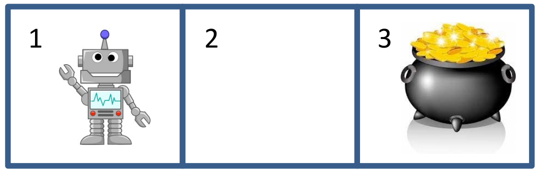
  - $(1, left, 0, 1)$
  - $\mathbf{(1, right, 0, 2)}$
  - $(2, left, 0, 2)$
  - $(2, left, 0, 1)$
  - $(1, left, 0, 1)$
  - $\mathbf{(1, right, 0, 1)}$
  - $\mathbf{(1, right, 0, 2)}$
  - $(2, left, 0, 2)$
  - $\mathbf{(2, right, 0, 3)}$
  - $\mathbf{(3, right, 1, 3)}$
  - $(3, left, 1, 2)$



$$P(s'|s, a = right) = \begin{bmatrix} 1/3 & 2/3 & 0 \\ 0 & 0 & 1/1 \\ 0 & 0 & 1/1 \end{bmatrix}$$

Model-based learning

- **Run around the environment and collect data points:**
  - $(1, left, 0, 1)$
  - $(1, right, 0, 2)$
  - $(2, left, 0, 2)$
  - $(2, left, 0, 1)$
  - $(1, left, 0, 1)$
  - $(1, right, 0, 1)$
  - $(1, right, 0, 2)$
  - $(2, left, 0, 2)$
  - $(2, right, 0, 3)$
  - $(3, right, 1, 3)$
  - $(3, left, 1, 2)$



Do we need more data points?

YES!

Model-based learning

```python
import numpy as np
np.set_printoptions(precision=2, suppress=True)

# States
S = ['1', '2', '3']

# Actions
A = ['L', 'R']

# Transition probabilities

L = np.array([[1.0, 0.0, 0.0],
              [0.8, 0.2, 0.0],
              [0.0, 0.8, 0.2]])

R = np.array([[0.2, 0.8, 0.0],
              [0.0, 0.2, 0.8],
              [0.0, 0.0, 1.0]])

P = [L, R]

# Reward function

R = np.array([[0.0, 0.0],
              [0.0, 0.0],
              [1.0, 1.0]])

gamma = 0.9
```

Model-based learning

```
STEPS = 10000

current_state = 0
num_actions = len(A)
num_states = len(S)

data_point = np.zeros((STEPS,4))

for t in range(STEPS):

    # choose action randomly
    action = np.random.choice(num_actions, p=[0.5,0.5])

    # choose next state
    next_state = np.random.choice(num_states, p=P[action][current_state, :])

    # obtain reward
    reward = R[current_state,action]

    # store data point
    data_point[t] = [current_state,action,reward,next_state]

    current_state = next_state
```

Model-based learning

```python
# initialize estimated transition probabilities
PestL = np.zeros((num_states, num_states))
PestR = np.zeros((num_states, num_states))
Pest = [PestL,PestR]

# initialize estimated reward
Rest = np.zeros((num_states, num_actions))

# initialize counters
st_act_counter = np.zeros((num_states, num_actions))
st_L_nst_counter = np.zeros((num_states, num_states))
st_R_nst_counter = np.zeros((num_states, num_states))
st_act_nst_counter = [st_L_nst_counter,st_R_nst_counter]

# compute counters and total reward
for t in range(STEPS):
    current_state, action, reward, next_state = data_point[t]

    st_act_counter[int(current_state),int(action)] += 1

    st_act_nst_counter[int(action)][int(current_state),int(next_state)] += 1

    Rest[int(current_state),int(action)] += reward


Pest[0] = st_act_nst_counter[0] / st_act_counter[:,0,None]
Pest[1] = st_act_nst_counter[1] / st_act_counter[:,1,None]
Rest = Rest / st_act_counter
```

Model-based learning

- **Result**
  - $\hat{P}(s'|s, a = left) = \begin{bmatrix} 1 & 0 & 0 \\ 0.81 & 0.19 & 0 \\ 0 & 0.8 & 0.2 \end{bmatrix}$
  - $\hat{P}(s'|s, a = right) = \begin{bmatrix} 0.2 & 0.8 & 0 \\ 0 & 0.2 & 0.8 \\ 0 & 0 & 1 \end{bmatrix}$
  - $\hat{R}(s, a) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$

And now we can use VI or PI!

Model-based learning

- To learn the model, we compute averages as follows:

$$\bar{x}_N = \frac{1}{N}\sum_{n=1}^{N} x_n$$

- For example, what is the average of $x_1 = 8, x_2 = 5, x_3 = 11$

$$\bar{x}_3 = \frac{8+5+11}{3} = 8$$

Model-based learning

- How to recompute the average with a new sample?

$$\bar{x}_{N+1} = \frac{\bar{x}_N \times N + x_{N+1}}{N+1}$$

- For example, what is the average of $x_1 = 8, x_2 = 5, x_3 = 11, x_4 = 3$

$$\bar{x}_4 = \frac{8 \times 3 + 3}{4} = 6.75$$

$$\bar{x}_4 = \frac{8 + 5 + 11 + 3}{4} = 6.75$$

Model-based learning

- Recomputing the average with a new sample

$$\bar{x}_{N+1} = \frac{\bar{x}_N \times N + x_{N+1}}{N+1}$$

$$\bar{x}_{N+1} = \frac{N}{N+1}\bar{x}_N + \frac{1}{N+1}x_{N+1}$$

$$\bar{x}_{N+1} = \frac{N+1-1}{N+1}\bar{x}_N + \frac{1}{N+1}x_{N+1}$$

$$\bar{x}_{N+1} = \left(1 - \frac{1}{N+1}\right)\bar{x}_N + \frac{1}{N+1}x_{N+1}$$

Model-based learning

$$\bar{x}_{N+1} = \bar{x}_N - \frac{1}{N+1}\bar{x}_N + \frac{1}{N+1}x_{N+1}$$

$$\bar{x}_{N+1} = \bar{x}_N + \frac{1}{N+1}(x_{N+1} - \bar{x}_N)$$

New estimate Previous estimate Step Error

Model-based learning

- Estimating the reward

$$\bar{r}_{t+1}(s_t, a_t) = \bar{r}_t(s_t, a_t) + \alpha_t\big(r_t - \bar{r}_t(s_t, a_t)\big)$$

- Where α_t can be $\frac{1}{N+1}$

Model-based learning

- Estimating the transition probabilities

Number of transitions from s to s' after selecting a

$$\bar{P}(s' \mid \mathrm{s}, \mathrm{a}) = \frac{N(s, a, s')}{N(s, a)}$$

Number of times a was selected in s

It's also an average!

Model-based learning

- Estimating the transition probabilities

$$\bar{P}(s' \mid \mathrm{s}, \mathrm{a}) = \frac{1}{N(s,a)} \sum_{t=1}^{N} \mathbb{I}(s_t = s, a_t = a, s_{t+1} = s')$$

It's also an average!

Model-based learning

- Estimating the transition probabilities

$$\bar{P}_{t+1}(s' \,|s_t, a_t) = \bar{P}_t(s' \,|s_t, a_t) + \alpha(\mathbb{I}(s_{t+1} = s') - \bar{P}_t(s' \,|s_t, a_t))$$

It's also an average!

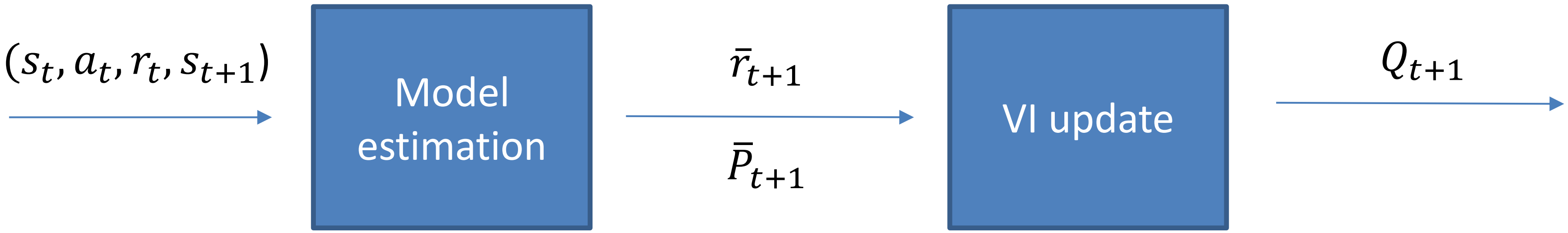
And now we can use VI or PI!

Model-based learning

- The model-based approach described **converges to the true parameters P and r** as long as **every state and action are visited infinitely often**

Model-based learning

- In practice, we interleave steps of model learning with steps of value/policy iteration



Model-based learning

- Given a sample (s_t, a_t, r_t, s_{t+1})

- Compute

$$\bar{P}_{t+1}(s' \,|s_t, a_t) = \bar{P}_t(s' \,|s_t, a_t) + \alpha(\mathbb{I}(s_{t+1} = s') - \bar{P}_t(s' \,|s_t, a_t))$$

$$\bar{r}_{t+1}(s_t, a_t) = \bar{r}_t(s_t, a_t) + \alpha_t\big(r_t - \bar{r}_t(s_t, a_t)\big)$$

- Compute

$$Q_{t+1}(s_t, a_t) \coloneqq \bar{r}_{t+1}(s_t, a_t) + \gamma \sum_{s' \epsilon S} \bar{P}_{t+1}(s' \,|s_t, a_t) \max_{a' \in A} Q_t(s', a')$$

Thank You

sardinha@inf.puc-rio.br