50.039 Theory and Practice of Deep Learning

W11-S3 More on Reinforcement Learning

Matthieu De Mari

Conclusion

1. What are **actor-critic** learning methods? And which problems do these approaches address?
2. What are more advanced problems in RL?
   - Markov states
   - Partially observable environment
   - SARSA
   - Non-stationary problems

A quick word on actor-critic methods (more advanced stuff, out of scope)

- In Deep Q-learning, we realized that most RL problems cannot have their Q functions computed easily.
- We then replaced the Q function with a Deep Neural Network to approximate this function.

A quick word on actor-critic methods (more advanced stuff, out of scope)

- In Deep Q-learning, we realized that most RL problems cannot have their Q functions computed easily.
- We then replaced the Q function with a Deep Neural Network to approximate this function.
- **Additional suggestion:** Can the reward function always be computed?

A quick word on actor-critic methods (more advanced stuff, out of scope)

- In Deep Q-learning, we realized that most RL problems cannot have their Q functions computed easily.
- We then replaced the Q function with a Deep Neural Network to approximate this function.
- **Additional suggestion:** Can the reward function always be computed?

→ **Replace more elements of the RL system with Deep Neural Networks.**

A quick word on actor-critic methods (more advanced stuff, out of scope)

Definition (actor-critic):

Actor-critic algorithms consist of two components.

- **Actor:** a DNN, whose purpose is to produce actions in response to given states, i.e. a policy. Can be trained as in Deep Q-learning.

A quick word on actor-critic methods (more advanced stuff, out of scope)

Definition (actor-critic):

Actor-critic algorithms consist of two components.

- **Actor:** a DNN, whose purpose is to produce actions in response to given states, i.e. a policy. Can be trained as in Deep Q-learning.
- **Critic:** a DNN, whose purpose is to evaluate the quality of the selected actions and suggesting directions for improvement, by defining a reward function or a Q function.

A quick word on actor-critic methods (more advanced stuff, out of scope)

Definition (actor-critic):

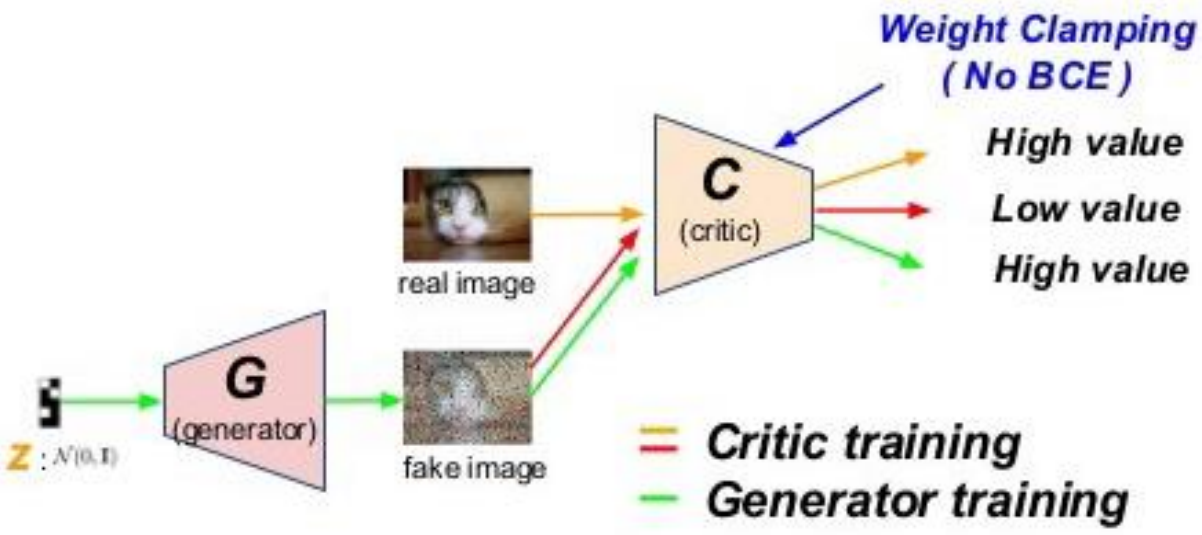
Actor-critic algorithms consist of two components.

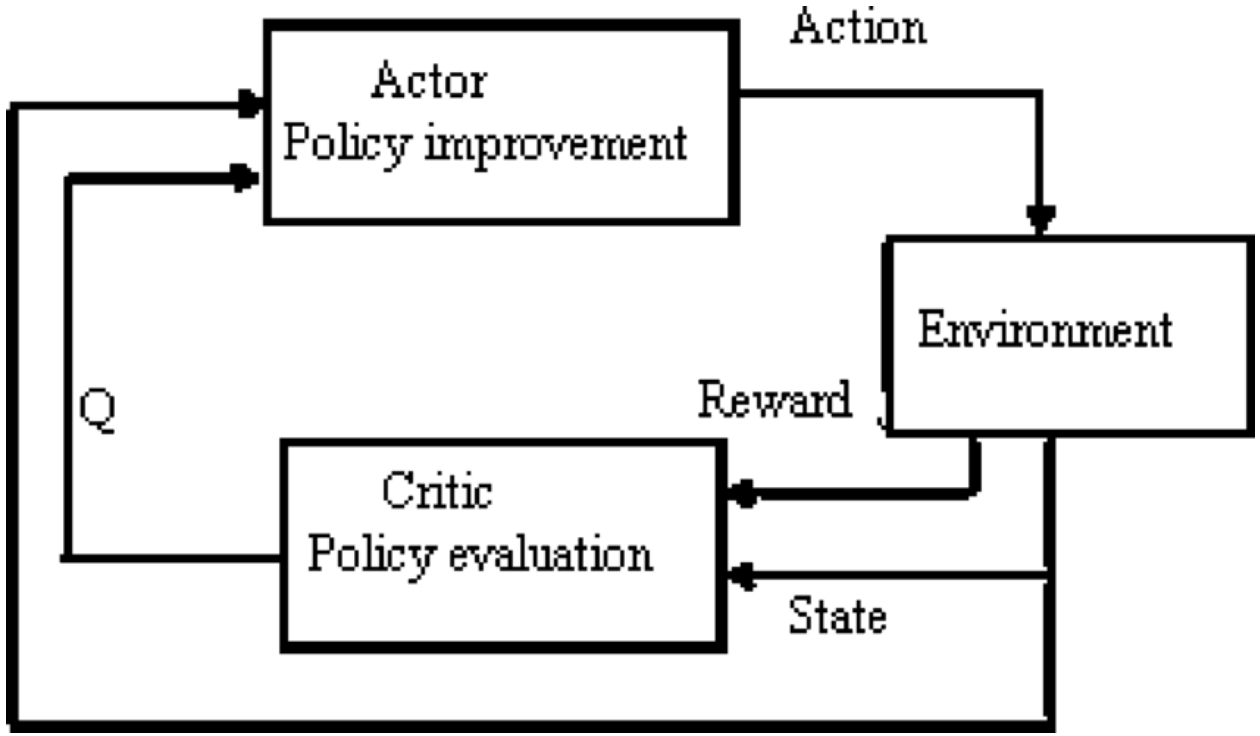
- **Actor:** a DNN, whose purpose is to produce actions in response to given states, i.e. a policy. Can be trained as in Deep Q-learning.
- **Critic:** a DNN, whose purpose is to evaluate the quality of the selected actions and suggesting directions for improvement, by defining a reward function or a Q function.

In a sense, **similar to the Generator-Critic pair** of the Wasserstein GANs!

- **Generator:** produce fake images
- **Critic:** evaluate said images

From Deep Q learning to actor-critic





RL with Human Feedback

Definition (Reinforcement Learning with Human Feedback):

Reinforcement Learning with Human Feedback (or RLHF) is a method in which an AI model (pre-trained or not), is typically represented as a reinforcement learning agent, and is fine-tuned based on feedback provided by humans to improve its performance on a specific task.

This approach combines the strengths of reinforcement learning algorithms, which learn through trial and error, with the expertise and judgment of human evaluators.

Markov Decision Processes (more advanced stuff, out of scope)

- Sometimes, the transition from state s_t to next state s_{t+1}, following from action a_t, will not always be **deterministic**.
- In that case, we have to define some system dynamics, as below.

$$p(s', r|s, a) = P(s_{t+1} = s', r_t = r \mid s_t = s, a_t = a)$$

- Similar to our previous problem, with a stochastic twist.
- It is called a Markov Decision Process (MDP) problem.

Markov Decision Processes (more advanced stuff, out of scope)

- In MDPs, all the previous formulas have to be reworked to account for the stochastic aspect of the problem.

$$V_t^{\pi}(s) = E[G_t \mid s_t = s]$$

$$Q_t^{\pi}(s, a) = E[G_t \mid s_t = s, a_t = a]$$

- In MDPs, the Q and V functions can still be learned from experience, but their Bellman equations change slightly, to account for the stochasticity.

Markov Decision Processes (more advanced stuff, out of scope)

- In MDPs, the Q and V functions can still be learned from experience, but their Bellman equations change slightly, to account for the stochasticity.

$$V_t^{\pi}(s) = E[G_t \mid s_t = s]$$

$$V_t^{\pi}(s) = E[R_t + \gamma G_{t+1} \mid s_t = s]$$

$$V_t^{\pi}(s) = \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma E[G_{t+1} \mid s_{t+1=s'}]\right]$$

$$\boldsymbol{V_t^{\pi}(s) = \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma V_{t+1}^{\pi}(s')\right]}$$

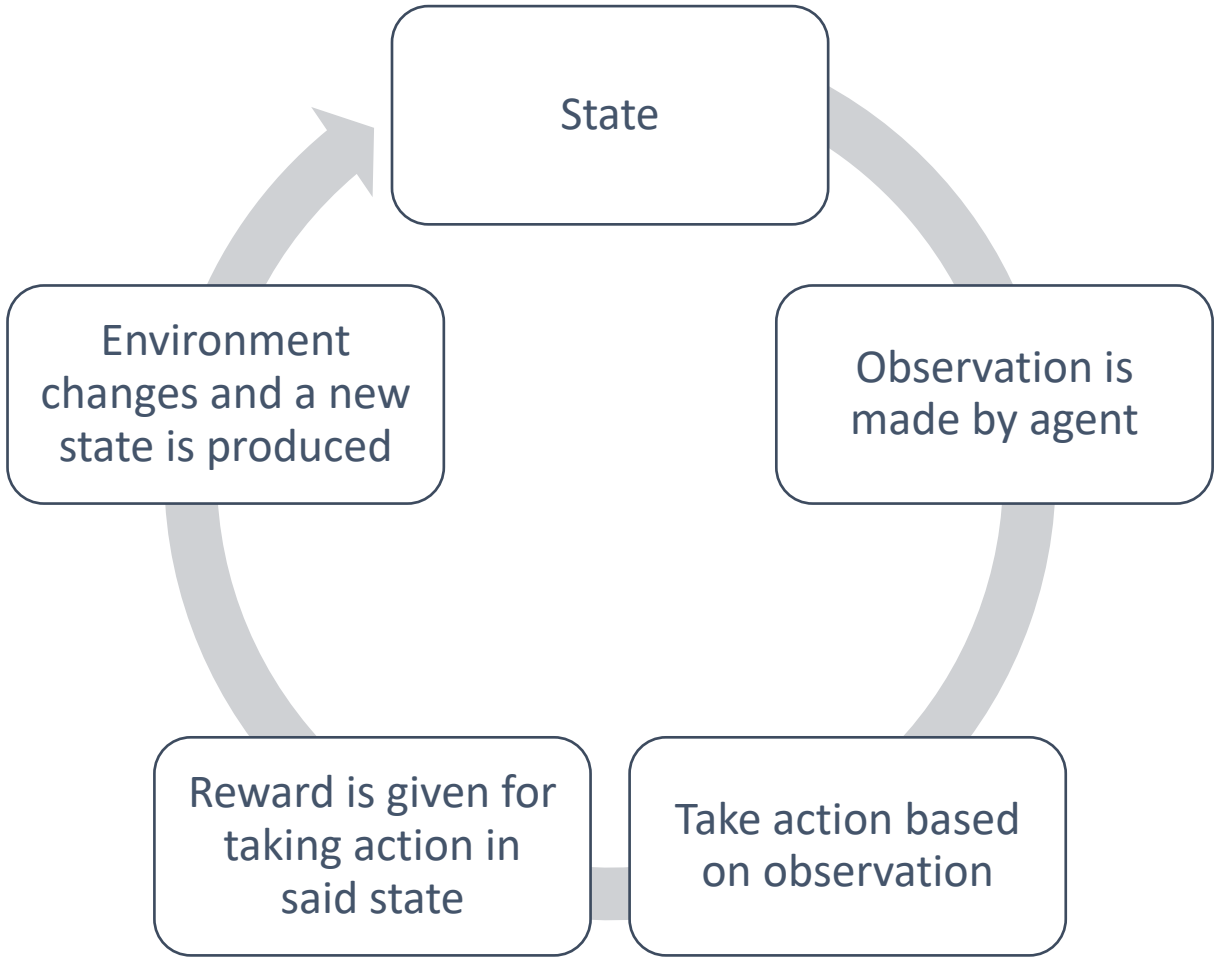
Partially observable MDP (more advanced stuff, out of scope)

Definition (partially observable Markov Decision Process):

In our original RL framework, we assumed that the agent was seeing the exact state of the game at each time t.

This is also an assumption, which can be challenged.

State s_t is ground truth, agent received observation o_t and uses to decide on action.



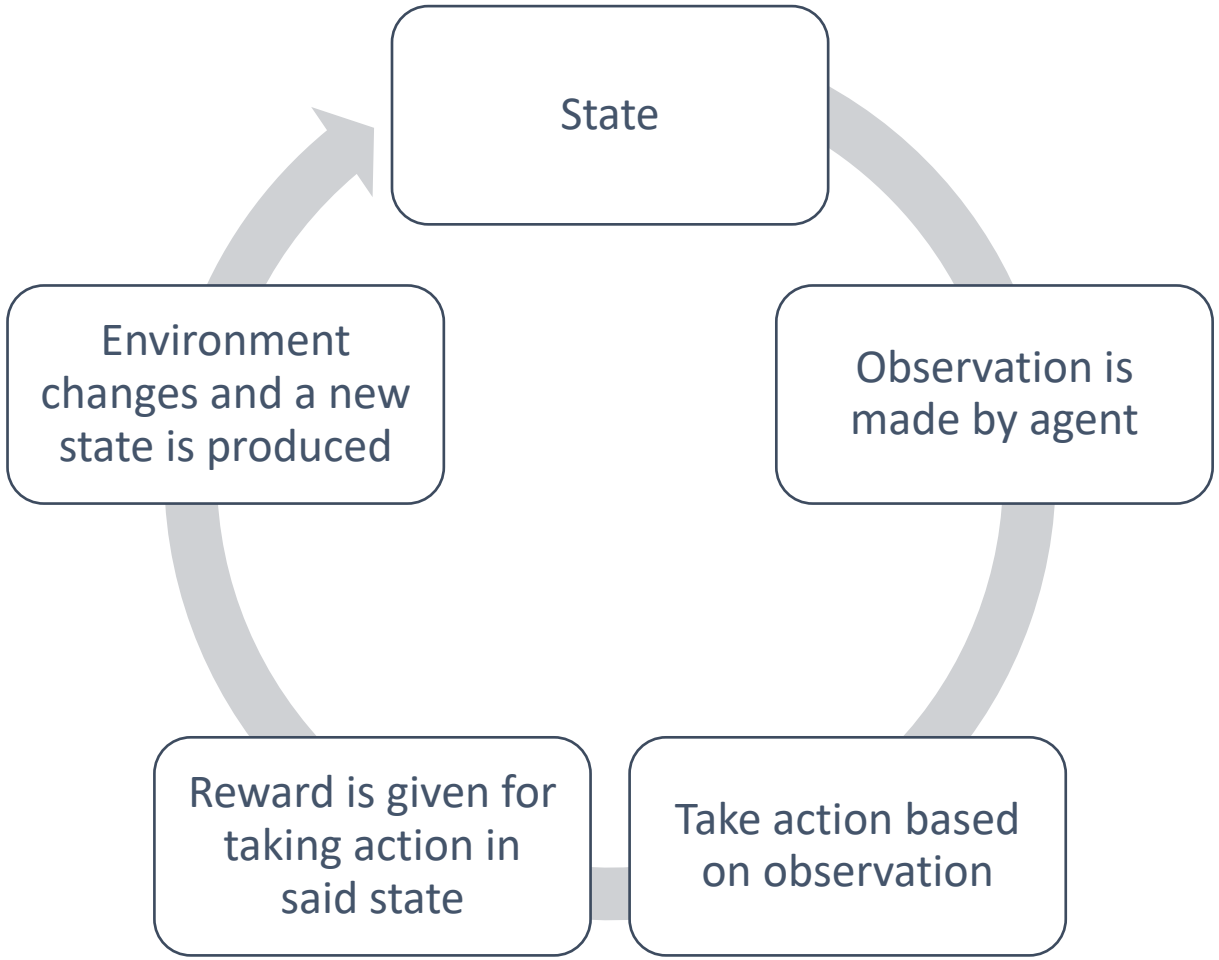
Partially observable MDP (more advanced stuff, out of scope)

Definition (partially observable Markov Decision Process):

The agent then decides on an observation made of the actual (partially hidden) state.

This is called a **partially observable Markov Decision Process**.

Agent will have to learn to observe on top of acting properly (e.g. card game, with opponent hiding his/her hand).

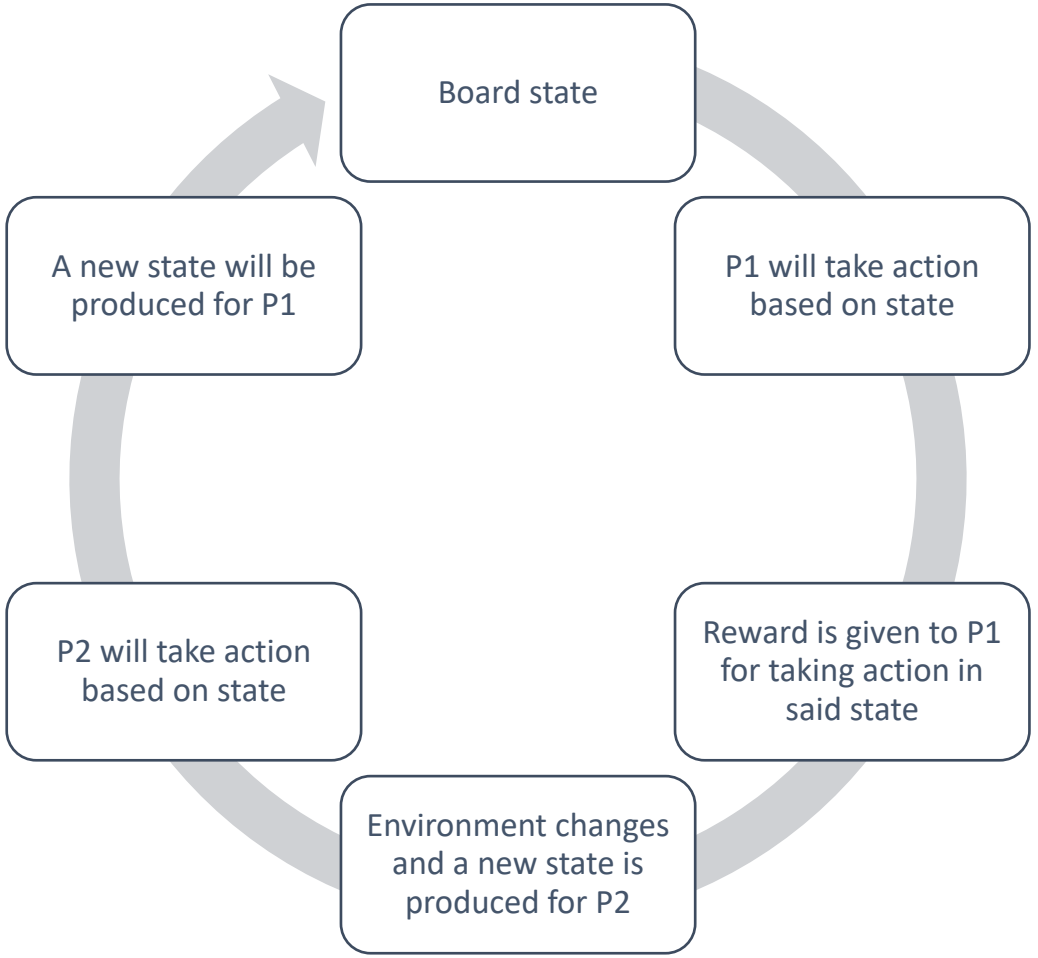


State-Action-Reward-State-Action or SARSA (more advanced stuff, out of scope)

- In many problems, e.g. Go, the new state seen by a given player is not the result of the action of the said player.
- Instead, another player has to act first, before a new state is produced.

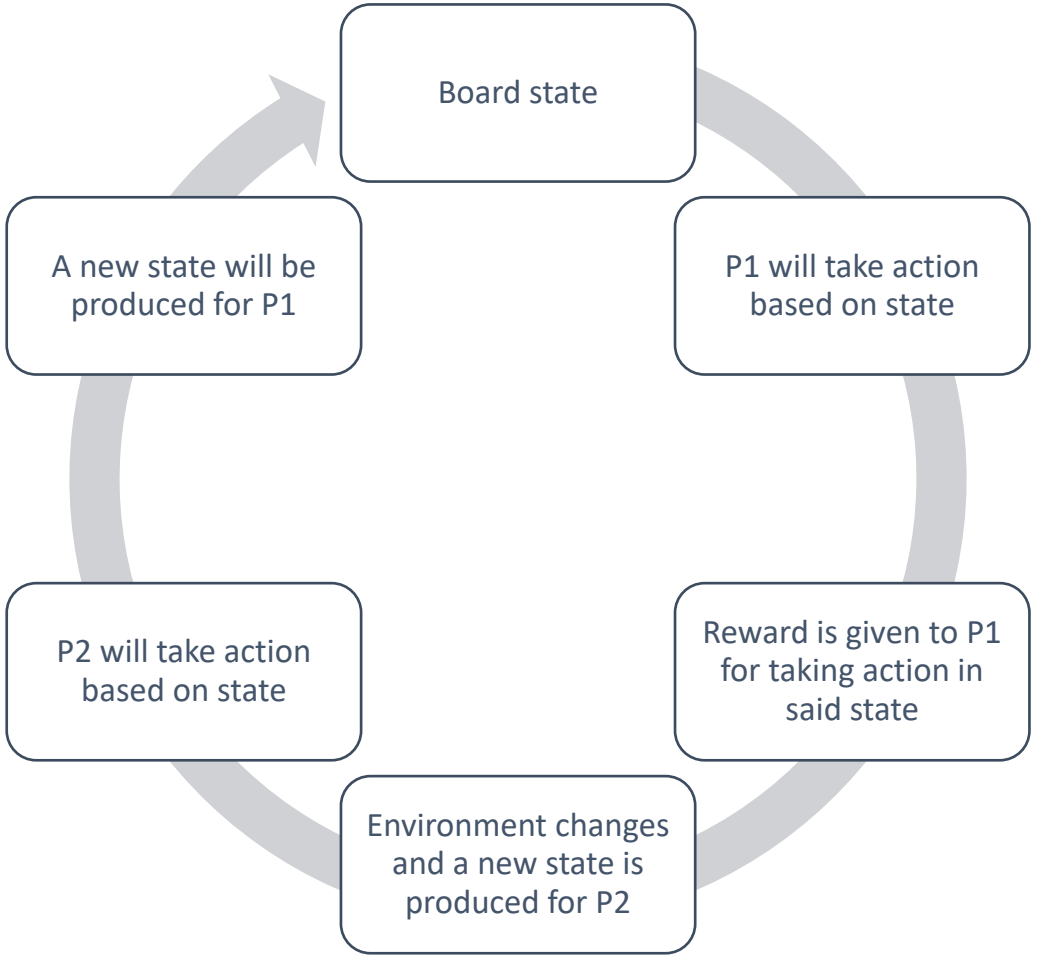
State-Action-Reward-State-Action or SARSA (more advanced stuff, out of scope)

- In many problems, e.g. Go, the new state seen by a given player is not the result of the action of the said player.
- Instead, another player has to act first, before a new state is produced.
- This adds steps to the cycle, which becomes (state, action, reward, state_P2, action_P2).



State-Action-Reward-State-Action or SARSA (more advanced stuff, out of scope)

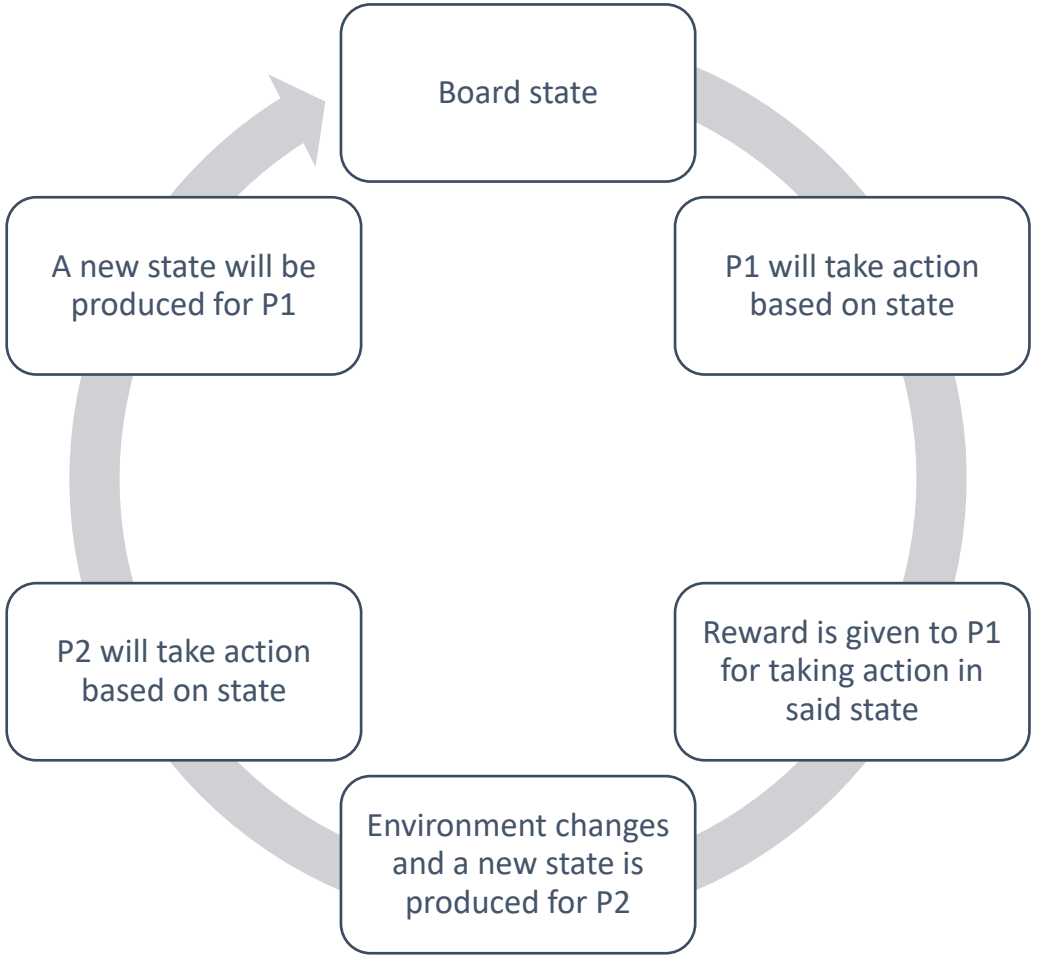
- This is called a **State-Action-Reward-State-Action (SARSA)** type of problem.
- In that case the agent has to learn how to play, but also has to learn how another player might respond to its actions.
- **RL meets game theory!**



State-Action-Reward-State-Action or SARSA (more advanced stuff, out of scope)

- This is called a **State-Action-Reward-State-Action (SARSA)** type of problem.
- In that case the agent has to learn how to play, but also has to learn how another player might respond to its actions.

→ Train two AIs at the same time (one for black, one for whites)? Make them play against each other and train together like generator-critic in WGANs?



Conclusion

1. What are **actor-critic** learning methods? And which problems do these approaches address?
2. What are more advanced problems in RL?
   - Markov states
   - Partially observable environment
   - SARSA
   - Non-stationary problems