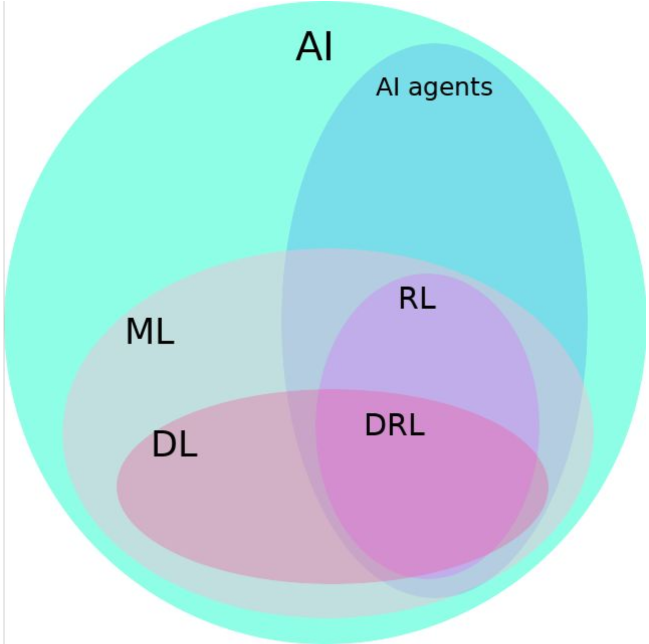
(Deep) Reinforcement Learning

Edu Vallejo, Miguel Ángel Gil, Pablo Felipe y Xabier Ugarte

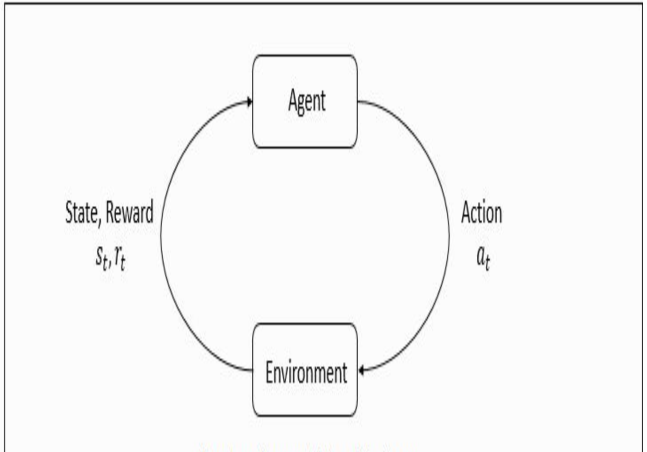
What is RL?

- RL: Applying machine learning techniques to automatically learn agents that do well (with respect to some metric) in an environment by optimizing for some objective function in the space of agent policies.
- Deep RL: augmenting the RL field with Deep Learning concepts.
  - deep architectures
  - methods to train the architectures

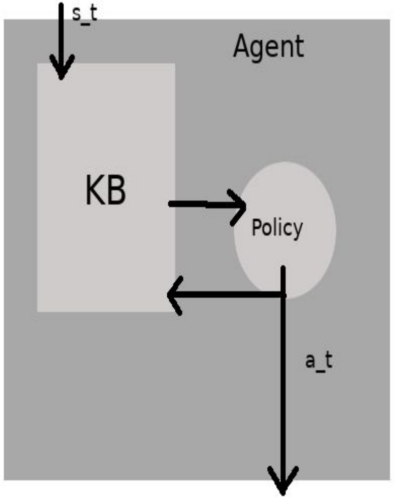


AI agents 101

- Environment
  - State space
  - State transition function
- Agent
  - Observation space
  - Action space
  - Knowledge base
    - Internal representation
    - Update functions
  - Policy
    - Internal representation
- Objective
  - Utility function
  - Fitness function



Agent-environment interaction loop.



Machine Learning in action

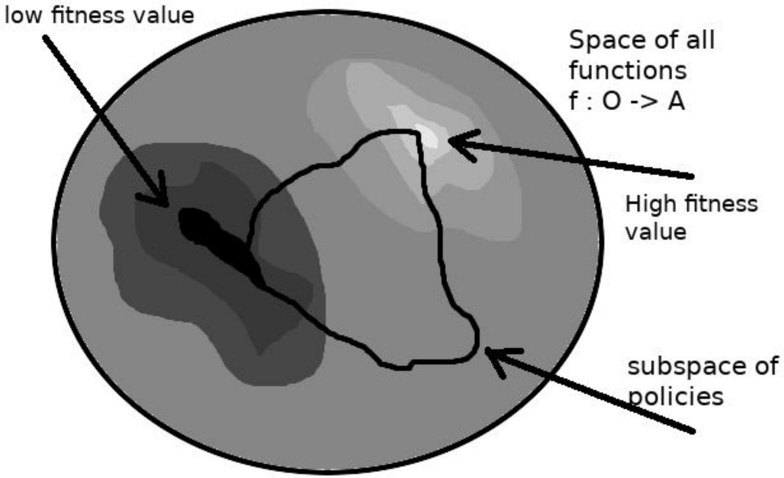
We have a measure of the “goodness” of the policy and we have defined a space of valid policies. We can formulate this as a search problem. Machine Learning techniques will help.

We have to restrict the search to subspaces with useful properties: Parametrized Policies

$$J(\pi_\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta}[R(\tau)]$$

$$\boxed{\nabla_\theta J(\pi_\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta}[\sum_{t=0}^{n} \nabla_\theta \log \pi(s_t|a_t) R(\tau)]}$$

$$\nabla_\theta J(\pi_\theta) \sim \nabla_\theta \hat{g} = \frac{1}{|D|} \sum_{\tau_i \in D} \sum_{t=0}^{n} \nabla_\theta \log \pi(s_t|a_t) R(\tau)$$



Enter reward

The utility function is a very weak supervision signal, too coarse-grain.

Reward is obtained every time-step, it informs about how good the previous action was (locally speaking).

The new utility function is the sum of the rewards.

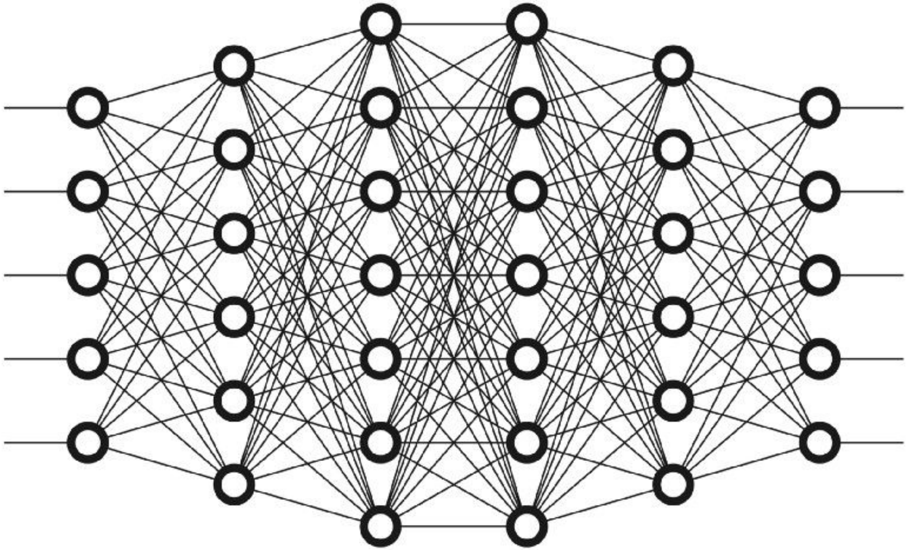
$$R(\tau) = \sum_{t=0}^{n} r_t$$

$$\boxed{\frac{1}{|D|} \sum_{\tau_i \in D} \sum_{t=0}^{n} \nabla_\theta \log \pi(s_t|a_t) \sum_{t'=t}^{n} r_{t'}}$$

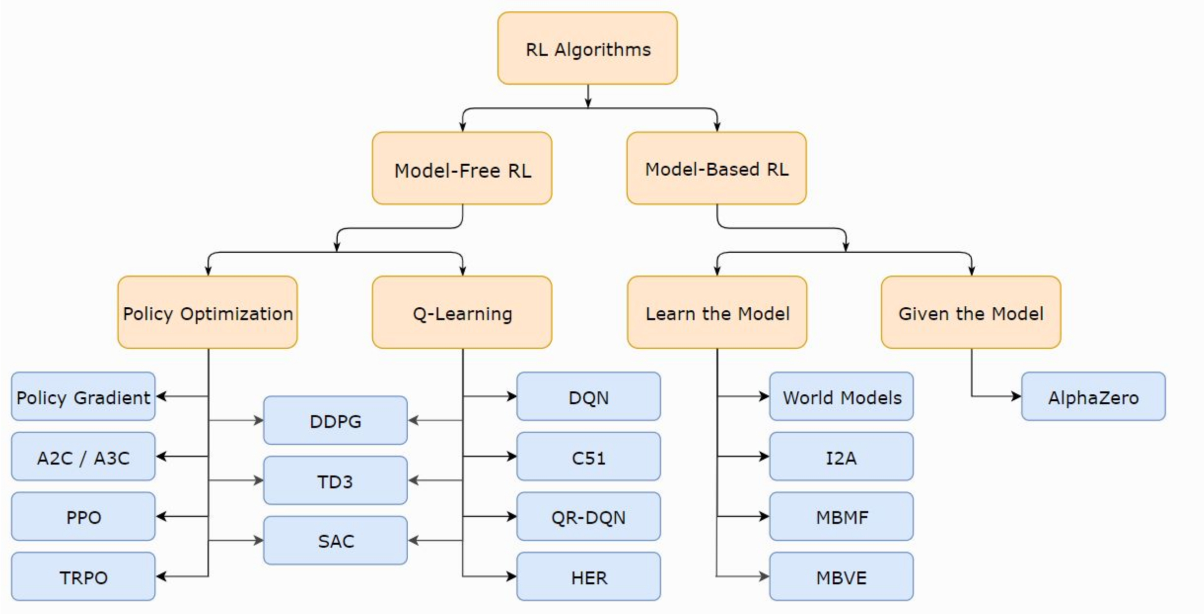
We need to go "Deep"

Use deep architectures for the internal representation of the parametrized policies:

- Differentiable policy
- Efficient gradient calculation (chain-rule)
- Efficient inference (vectorized, GPUs)
- Very high capacity model



Taxonomy of RL Algorithms



Model-Free vs Model-Based

The model predicts state transitions and rewards.

- Capability to think ahead and plan
- Bias in the model can be exploited by the agent

Model-free algorithms are more popular because they easier to implement

Model-Free Algorithms

Policy Optimization

- Deterministic policy
- Stochastic policy

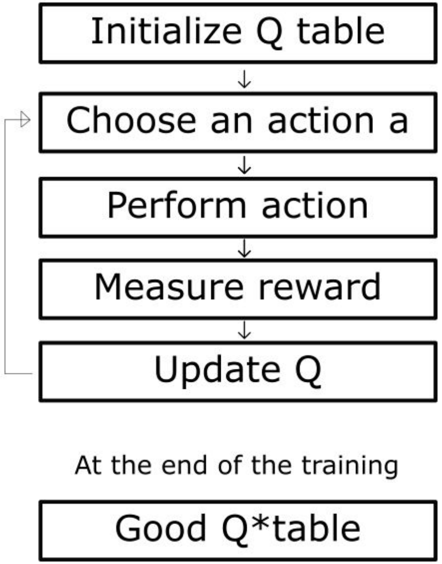
Mostly performed on-policy

Model-Free Algorithms

Q-Learning

Learns the action-value function $Q(s, a)$

Typically performed off-policy



Model-Free Algorithms

Policy Optimization vs Q-Learning

- Policy optimization algorithms: More stable and reliable

- Q-learning: More sample efficient because they reuse data more effectively

Model-Free Algorithms

Hybrid Algorithms

These algorithms aim to combine the strengths of Q-learning and policy gradients.

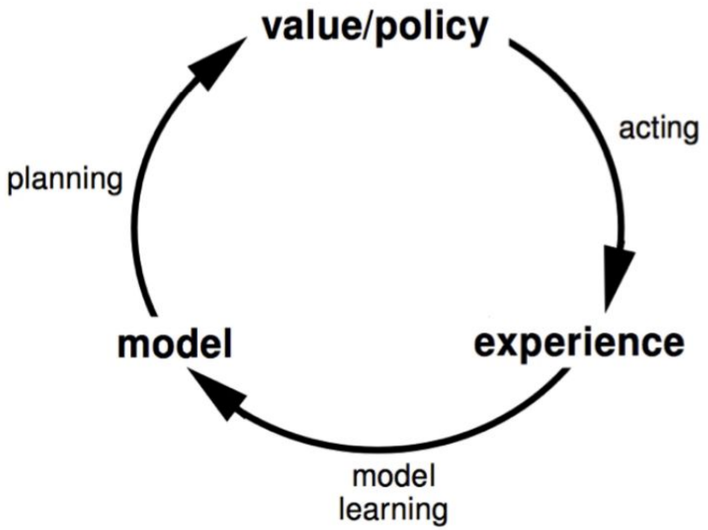
Model-Based Algorithms

Learn the Model

1. Run a base policy
2. Observe the trajectory
3. Fit the model

Given the Model

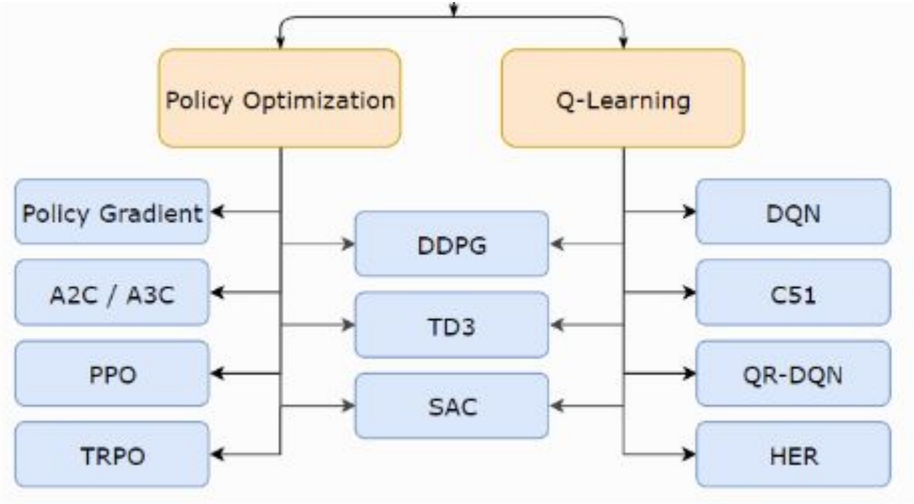
There are cases where some rules define the model (e.g., Go)



Algorithms

Main focus:

- Policy Optimization
- Policy Optimization + Q-Learning



Vanilla Policy Gradient

Goal: train **stochastic** policies in an **on-policy** way (no replay buffer, less sample efficient)

It can be used in discrete and continuous environments

- **Raise** the probability of choosing actions that generate **good results**
- **Lower** the probability of choosing actions that generate **bad results**

Proximal Policy Optimization

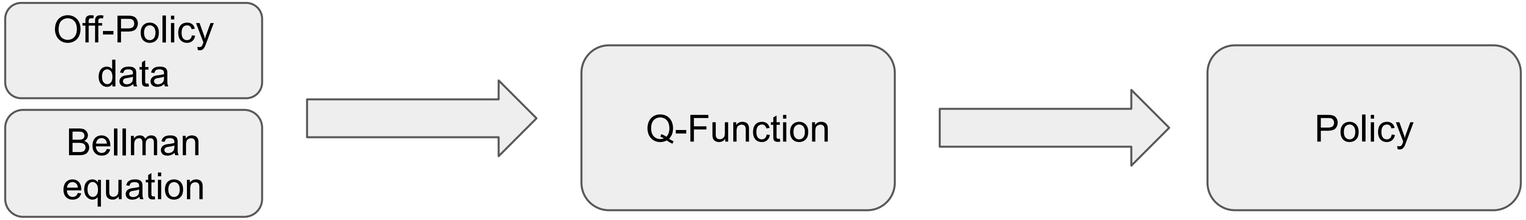
Motivation: to avoid getting too far from the previous policy when taking big steps of improvement

Two primary variants of PPO:

- PPO-Penalty: it uses KL-constraint previously proposed by Trust Region Policy Optimization (TRPO)
- PPO-Clip: it relies on specialized clipping in the objective function

Interpolation between Policy Opt. and Q-Learning

- Deep Deterministic Policy Gradient (DDPG)



- Soft Actor Critic (SAC)
  - Bridge between stochastic policy optimization and DDPGlike approaches

Agent experience

- Exploitation
- Exploration

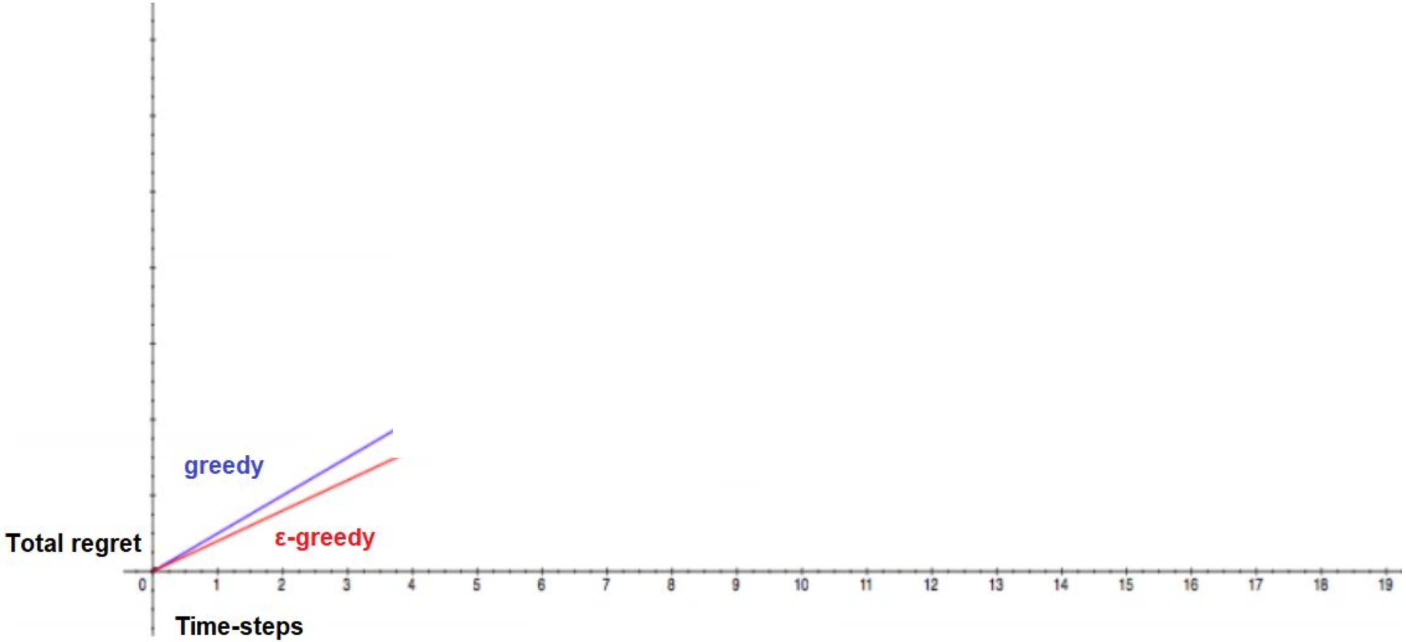
Exploration vs. Exploitation

- Batch of data?
- Data gathering?
- Agent exploration or Agent exploitation?

Regret in Reinforcement Learning - Notion of regret

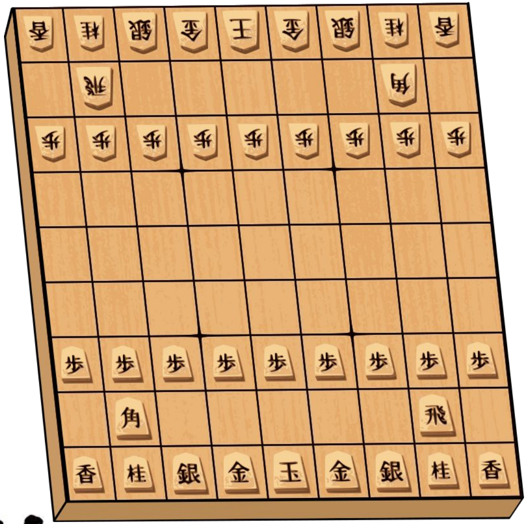
Greedy

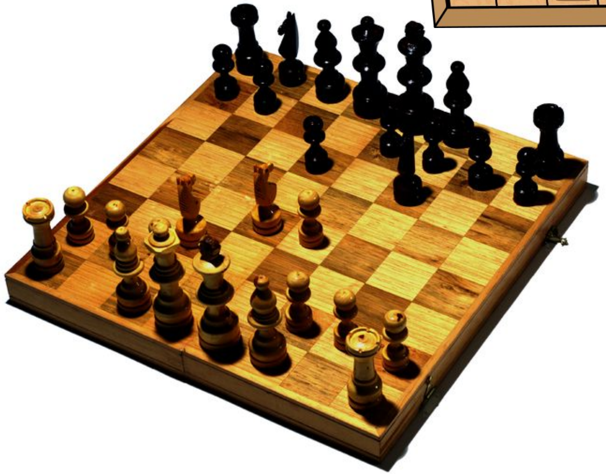
Epsilon Greedy



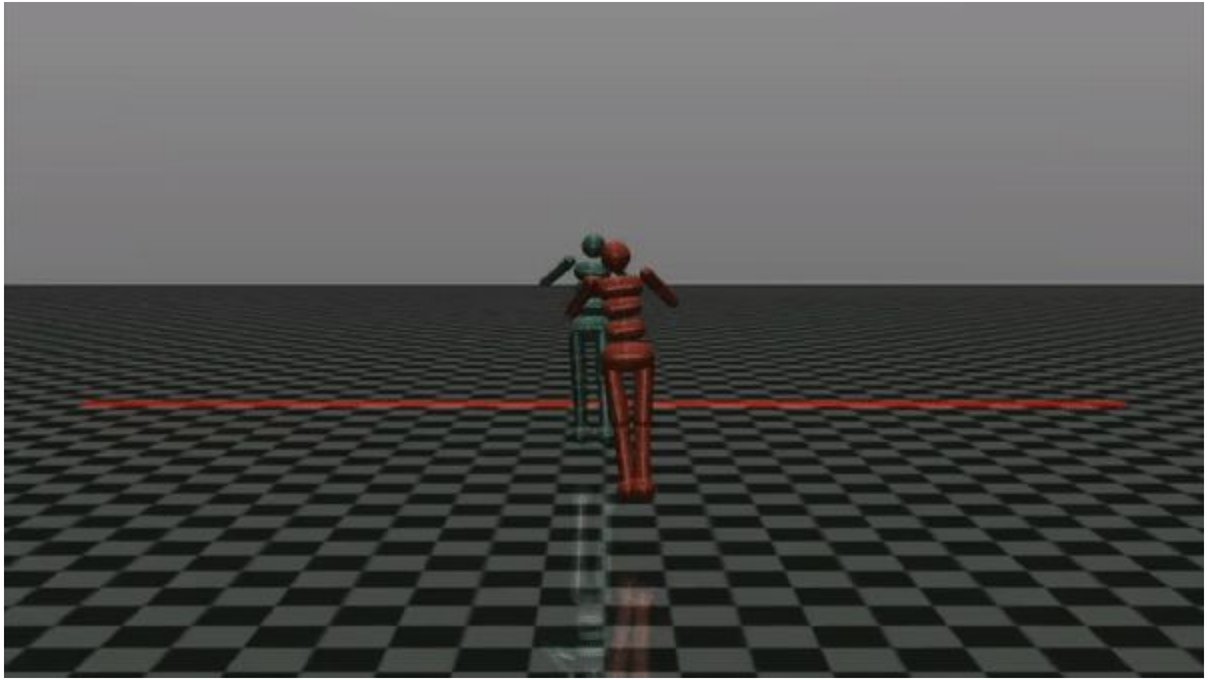
Self-Play

- Example: **AlphaZero**

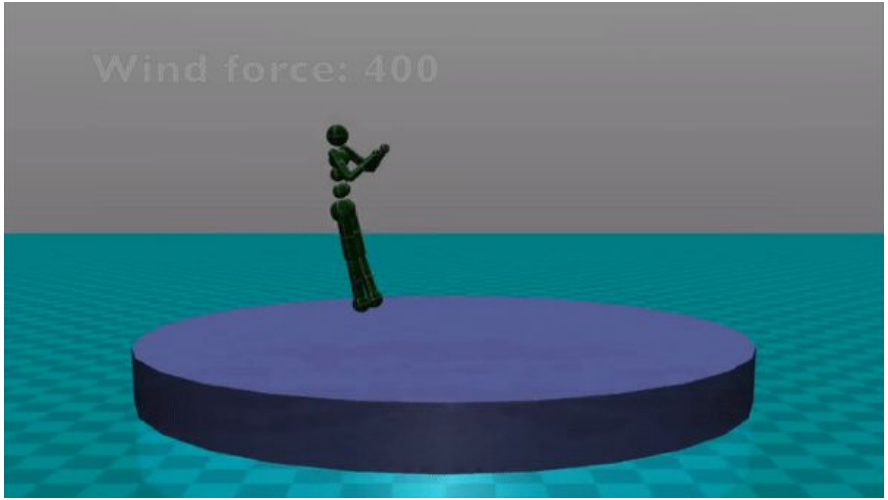


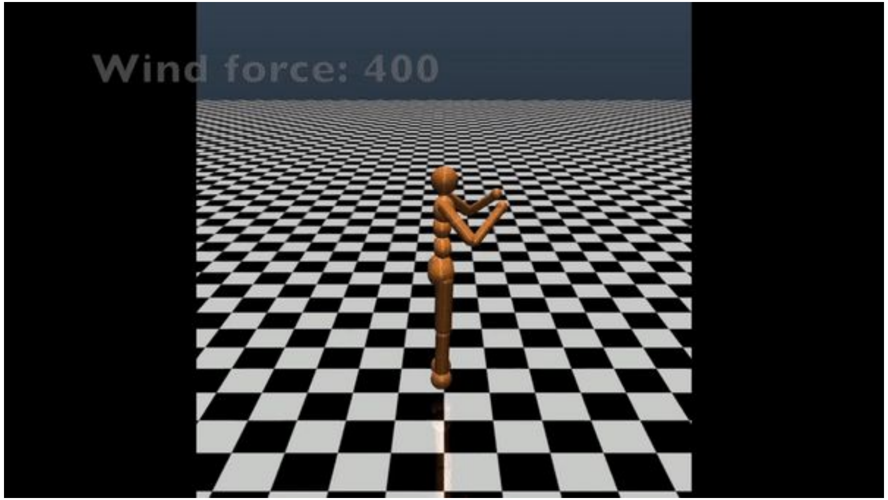


Competitive self-play



Difference between classical RL and self-play





Conclusions

- RL is a very powerful tool
- A lot of potential
- The future of AI