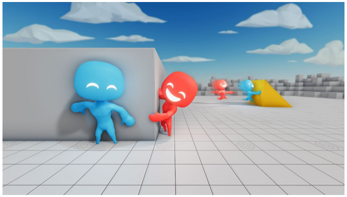
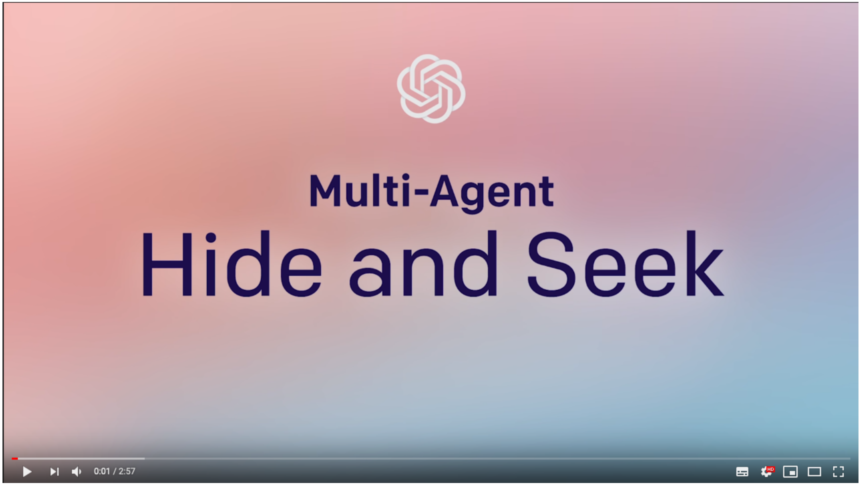
Emergent Tool Use from Multi-Agent Interaction
https://openai.com/blog/emergent-tool-use/

Pascal Leroy

April 2020



Teaser



https://www.youtube.com/watch?v=kopoLzvh5jY

Content

RL basics: The environment

Markov decision process (MDP)

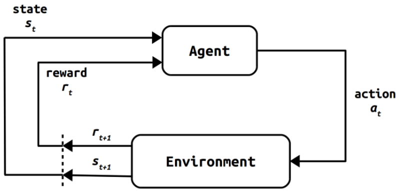


Figure: RL environment [1]

Composed of:

- A set of states $s \in \mathcal{S}$
- A set of action $a \in \mathcal{A}$
- Transition function: $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$
- Reward function: $r_t = R(s_{t+1}, s_t, a_t)$
- Policy: $\pi(a_t|s_t)$

The agent **goal**: To maximize its total expected sum of (discounted) rewards $\sum_{t=0}^{\infty} \gamma^t r_t$ with $\gamma \in [0, 1)$, obtained with the optimal policy π^*.

[1]Richard S. Sutton and Andrew G. Barto. 2018. Reinforcement Learning: An Introduction. A Bradford Book, Cambridge, MA, USA.

RL basics: Paradigms

Two main learning paradigms in RL:

- Model based: to learn a model of the transition and reward functions and use this model to find the best policy.
- Model free: to learn directly the policy without modelling the environment.

Two main families in model free:

- Value based: to predict the value of each action and take the best one.
- Policy based: to compute the probabilities of taking each action.

RL basics: Value Based

Value Based

- State Value of a policy π:

$$V^\pi(s_t) = \mathbb{E}_\pi \left[r_t + \gamma V^\pi(s_{t+1}) | s_t \right]$$

- State-Action Value of a policy π:

$$Q^\pi(s_t, a_t) = \mathbb{E}_\pi \left[r_t + \gamma Q^\pi(s_{t+1}, a_{t+1}) | s_t, a_t \right]$$

- The optimal policy is:

$$\pi^*(s_t) = \operatorname*{argmax}_a Q^{\pi^*}(s_t, a)$$

RL basics: Value Based

- The optimal value functions obey the Bellman equation:

$$V(s_t) = \max_a \sum_{s_{t+1}} P(s_{t+1}|s_t, a)(r_t + \gamma V(s_{t+1}))$$

$$\begin{aligned} Q(s_t, a_t) &= \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t)(r_t + \gamma V(s_{t+1})) \\ &= \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t)(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})) \end{aligned}$$

Note that we consider that the agent takes optimal action.

RL basics: Value Based

Tabular Q-Learning and Q-value iteration: find optimal Q-values.

- By running the environment, obtain transitions (s_t, a_t, r_t, s_{t+1}) where s_{t+1} is the next state after taking the action a_t in state s_t. r_t is the reward of this transition.
- Store the Q-values in a table (one cell for each state-ction pair), initialize these to 0.
- For each transition, update the table:

$$Q_{k+1}(s_t, a_t) = (1-\alpha)Q_k(s_t, a_t) + \alpha \left(r_t + \gamma \max_a Q_k(s_{t+1}, a) \right)$$

  α being the learning rate.
- Problem? Intractable if state space or action space is very large
- $\rightarrow$ Approximate Q-values with neural network.

RL basics: Deep Q-Learning

- **Replay buffer**: It is a collection of transitions (s_t, a_t, r_t, s_{t+1}).
- Sampling N transitions allows to update the network, minimizing the following loss:

$$\mathcal{L}(\theta) = \sum_{i=0}^{N} [r_i + \gamma \max_{a'} Q(s_{i+1}, a'; \theta') - Q(s_i, a_i; \theta)]^2$$

- θ' denotes the parameters of the **target network**, a copy of θ that is periodically updated.
- Ideally, transition samples should not be consecutive in order to not bias the update.

RL basics: Policy based

Policy based

- We denote
  - $\pi_\theta(a_t|s_t)$, the policy defined by a neural network with parameters θ.
  - $\tau = (s_0, a_0, r_0, s_1, a_1, .., s_T)$, a trajectory.
  - $r(\tau) = \sum_t r_t$, the total reward in a trajectory.
- **Goal**: Maximise $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta} r(\tau)$
- **How?** Gradient ascent!

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta_t)$$

- Skipping the math:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[r(\tau)\left(\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t, s_t)\right)\right]$$

RL basics: Reinforce and baseline

- Use the discounted sum of reward $G_t(\tau) = \sum_{j=0}^{T} \gamma^j r_{t+j}$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\left(\sum_t^T G_t(\tau) \nabla_\theta \log \pi_\theta(a_t, s_t) \right) \right]$$

- This is called REINFORCE
- Smaller variance with a baseline:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\left(\sum_t^T (G_t(\tau) - b) \nabla_\theta \log \pi_\theta(a_t, s_t) \right) \right]$$

- A good example of baseline is the average reward.

RL basics: Q Actor-Critic

- Forget about the baseline for now.
- Value-function reminder:

$$Q(s_t, a_t) = \mathbb{E}_{(r_t, s_{t+1}, \ldots, s_T)} \left[G_t(\tau) \right]$$

- Back to our gradient, it is possible to approximate the Q-function with a set of parameter w.

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_{\tau \sim \pi_\theta} \left[\left(\sum_t^T G_t(\tau) \nabla_\theta \log \pi_\theta(a_t, s_t) \right) \right] \\ &= \mathbb{E}_{\tau \sim \pi_\theta} \left[\left(\sum_t^T Q_w(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t, s_t) \right) \right] \end{aligned}$$

- $\rightarrow$ Q Actor-Critic: In addition,

RL basics: Advantage Actor-Critic

Back to the baseline:

- What about a baseline that depends ONLY on the state?
- $\rightarrow$ Learn the Value function to be used as the baseline!
- Advantage value:

$$\begin{aligned} A(s_t, a_t) &= Q_w(s_t, a_t) - V_v(s_t) \\ &= r_t + \gamma V_v(s_{t+1}) - V_v(s_t) \end{aligned}$$

- Actor: learns the policy, Critic: learns the advantage

$$\longrightarrow \nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\left(\sum_t^T A_v(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t, s_t) \right) \right]$$

RL basics: Improvements

Two improvements of interest:

- PPO: Proximal Policy Optimization
  - Avoid large policy updates by penalizing large changes of the policy.

- GAE: Generalised Advantage Estimator
  - Compute an advantage as a sum of temporal differences of a future time horizon.
  - Improve the sample efficiency of policy gradient methods.

Summary

- We can use a neural network to approximate the values of actions in order to choose the best one.
- We can use a neural network to parametrize a policy, which is updated using policy gradient toward the optimal policy.
- It is possible to learn both the policy and the value of actions taken by this policy with the Actor-Critic.
- There are techniques that reduce variance in the policy updates.

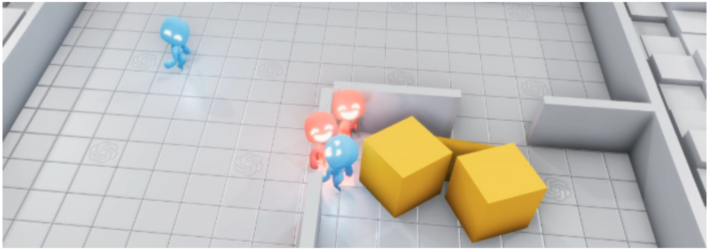
Multi-Agent environment

- Partially observable multi-agent environment $\rightarrow$ ~~MDP~~ but **Markov Game**
- N agents
- Set of state $\mathcal{S}$
- Set of all agents observations: $\mathcal{O}^1, ..., \mathcal{O}^N$
- Set of all agents actions: $\mathcal{A}^1, ..., \mathcal{A}^N$
- Transition function: $\mathcal{S} \times \mathcal{A}^1 ... \mathcal{A}^N \rightarrow \mathcal{S}$ (can still be a distribution)
- Rewards: $\mathcal{S} \times \mathcal{A}^1 ... \mathcal{A}^N \rightarrow \mathbb{R}^N$
- The goal of each agent remains the same: maximize its own (discounted) reward.

Hide and Seek environment

- Hiders (blue) trie to avoid line of sight of the Seekers (red).
- Objects that can be grab or lock.
- Preparation phase: only the hiders can perform action in the beginning.

- Team based reward
- Hiders: +1 if hidden, -1 if seen.
- Seekers: opposite.
- 0 if no one is seen by the seekers.
- Time limit: 240



Hide and Seek environment

- Action space:
  - Move: discretized forces along x and y axis and torque around their z axis.
  - Grab or lock the closest object. 2 binaries. (Only object in front of them within a small range)
    - Grab: the object is bound to the agent while boolean is True.
    - Lock: the object is locked and cannot be moved. Unlocking is available if the agent is part of the agent team that has locked the object.
- Observation space:
  - Position, velocity and size(or objects), in a 135 degree cone in front of the agent.
  - "LIDAR": 30 range sensors around the agent.
- Go to the blog!

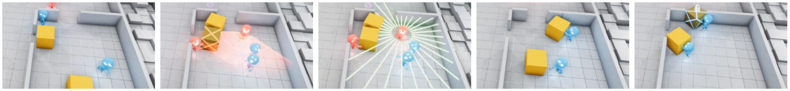
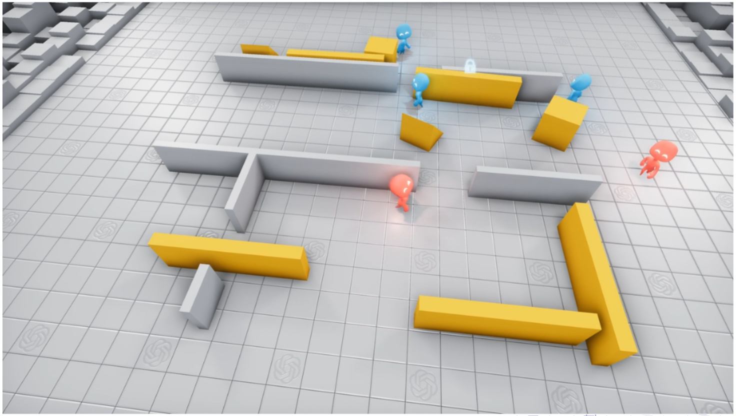


Figure: `https://openai.com/blog/emergent-tool-use/`

Hide and Seek scenarios

Normal environment:

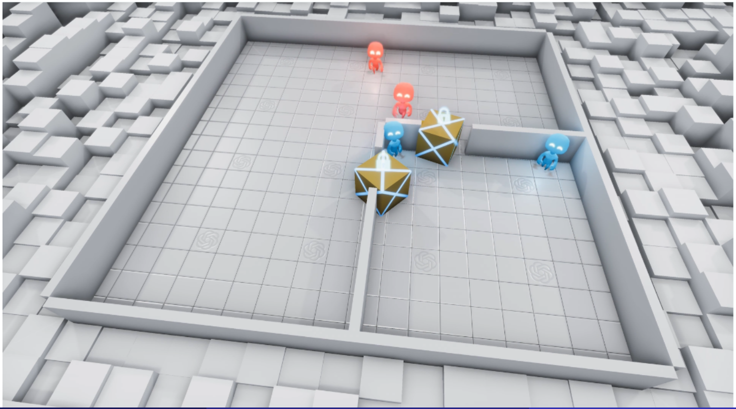
- 1 to 3 hiders, 1 to 3 seekers
- 3 to 9 boxes (at least 3 elongated)
- 2 movable ramps
- random walls and rooms
- random initial position



Hide and Seek scenarios

Quadrant environment:

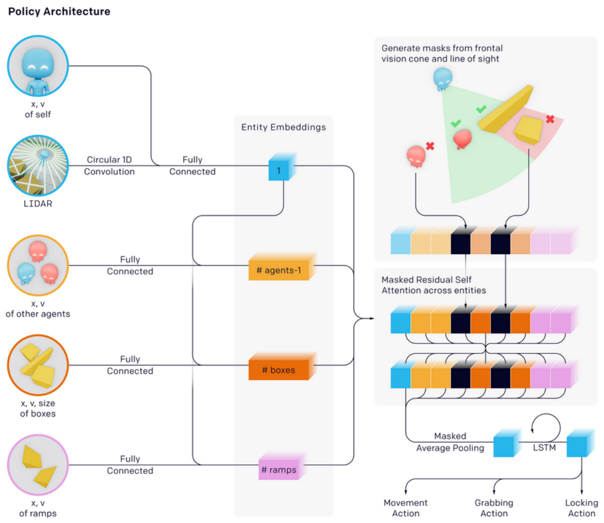
- 2 hiders, 2 seekers
- 2 boxes inside the room
- Seekers spawn outside the room
- 1 movable ramp that cannot be blocked
- 1 fixed room with 1 or 2 random doors



Hide and Seek optimization

- Actor-Critic $\rightarrow$ 2 sets of parameters.
- The actor network, optimised with PPO and GAE, produces an action distribution based on the agent observations.
- The critic network that predicts the discounted future returns.
- Use centralize training, decentralized execution (CTDE).
  - At training time: the critic has access to the full state to lean a centralized value function.
  - At execution time: the policy network is used normally.
- Self-play: each agent acts independently but share the same network parameters.
- 5% chance of using a past policy version to improve robustness.

Hide and Seek architecture



Hide and Seek entity-centric observations

- The observation is composed of different entities: agents, boxes, ramps.
- **The first step** is to embed each entity. Same objects are embedded with the same parameters.
- Self information: the concatenation of Lidar (that went through a circular 1-D convolution) with position and velocity of the agent are embedded with a fully connected layer.
- There are three other entities. Each one is concatenated with the embed of the self information before their own embed:
  1. Other agents information $(N-1)$: position and velociity.
  2. Box information (Number of boxes): position, velocity and size.
  3. Ramp information (Number of ramps): position and velocity.

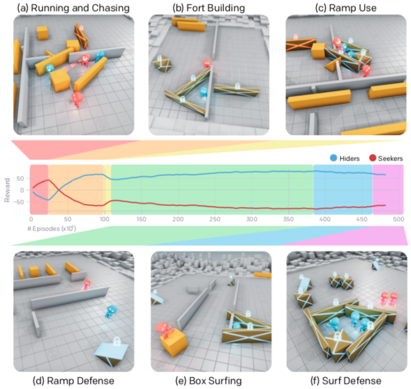
  The number of these entities varies depending on the scenario.

Hide and Seek Policy optimization

- **The second step** is to pass all embedded entities through a residual self-attention block (unobservable entities are masked away).
- Attention mechanisms allow to capture object-level information.
- "A self-attention module takes in n inputs, and returns n outputs. The self-attention mechanism allows the inputs to interact with each other ("self") and find out who they should pay more attention to ("attention"). The outputs are aggregates of these interactions and attention scores."
- **The third step** is to concatenate the average-pool entities embedding with self information.
- **The fourth step** is to pass through a LSTM.
- **The last step** is to pass through the three sperate heads (one for each action type)
- Full parameter values are detailed in the paper.

Hide and Seek Emergent behavior

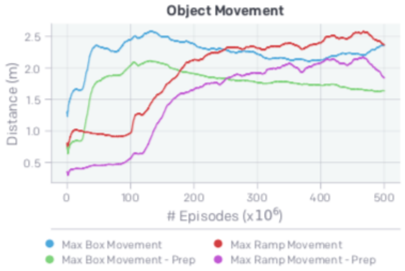
- In both scenarios, different strategies emerge as the agents train $\rightarrow$ autocurriculum. They can be seen on the blog.
- In the paper, they insist on the fact that there is no incentives for agents to interact with objects, this is only a result of autocurriculum induced by the competition.

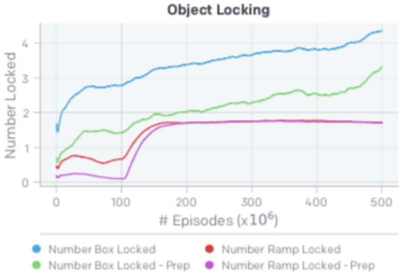


Hide and Seek Emergent behavior

Tool interaction

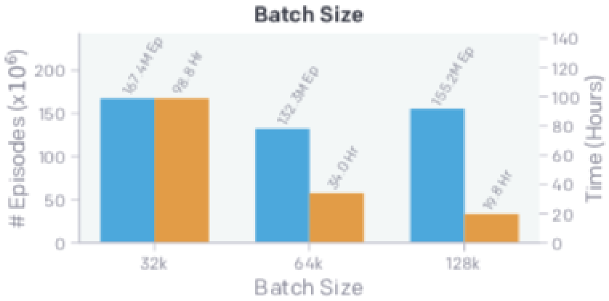
- Agents learn to divide the labor: Against box surfing, 2 and 3 hiders lock 25% and 36% more boxes than a single hider.
- It is possible to track the stages of emergent policy with the interaction of tools in the environment.





Hide and Seek Emergent behavior

- Scale matters!
- In the Figure, the number of episode (blue) and time (orange) to reach stage 4 (ramp defense) in term of batch size.
- The size of batch size is the number of transition chunks.
- The model uses 1.6 million parameters.



Hide and Seek Evaluation

- Reward in Multi-Agent is not sufficient. Are agents improving evenly or have they stagnated?
- ELO score (or Trueskill) allows to establish a ranking and measure improvement compared to other policies. However, it does not differentiate adaptation and improvement of already learned skills.
- Their propose two evaluations scheme.
  1. Comparison to intrinsic motivation.
  2. Inteligence tests: transfer and fine-tuning.

Hide and Seek Intrinsic motivation

- **Goal?** Compare behavior learned in Hide and Seek with common unsupervised exploration techniques.
- Underlying idea: Can agent behave like human with these unsupervised exploration methods.
- Count based exploration: the agent receives reward when visiting states that has not been visited much.
- Very dependent of state representation.
- **Result**: The learned behavior is very not human-like.
- See blog results.

Hide and Seek Intelligence tests

- **Goal?** Use transfer learning to evaluate network parameters on five new tasks.
- Three configuration:
  1. Trained from scratch
  2. Pre-trained with Hide and Seek and fine tuned.
  3. Pre-trained with count based and fine tuned.
- These tests include supervised learning and reinforcement learning but are single-agent:
  1. Object counting
  2. Lock and return
  3. Sequential lock
  4. Construction from blueprint
  5. Shelter construction
- Notes:
  - Same spaces and observation (fake hiders)
  - Object counting task: the action heads are replaced by a classification head (7-classes representing whether 0 through 6 boxes have gone to the left)

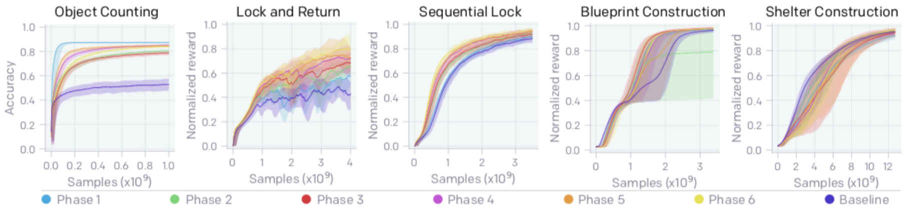
Hide and Seek Intelligence tests

Results:

- Pre-trained with Hide and Seek configuration is better on Lock and return, Sequential lock and Construction from blueprint.
- Pre-trained with count-based configuration is better on Object counting.
- Pre-trained with Hide and Seek configuration achieves the same results as the trained from scratch configuration on Shelter construction but sligthly slower.

More intelligence tests

Intelligence tests are performed at different phases of emergence.



- Improves on navigation and memory as it progresses.
- Object counting is transient.
- Performance on manipulation tasks (constructions) is uncorrelated to the phases. The most surprising: policy from phase 1 performs comparably well to others.

Alternative games

Hide and Seek with food reward.

- Can eat food if:
    - after preparation phase.
    - hiders not seen by seekers.
    - hider close and visible to the food.

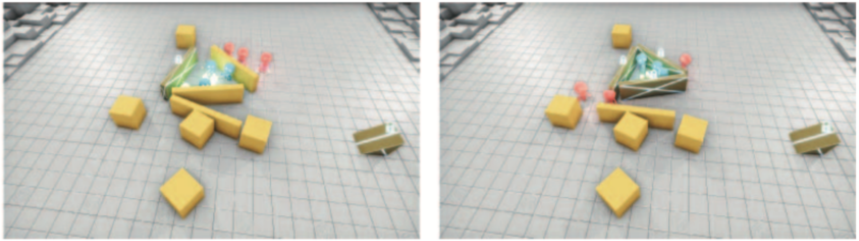


Figure: Hide and Seek with food reward.

Alternative games

Hide and Seek with food reward: Results.

- Incentivized to build forts around food.
- Four levels of skill progression (see Figure)

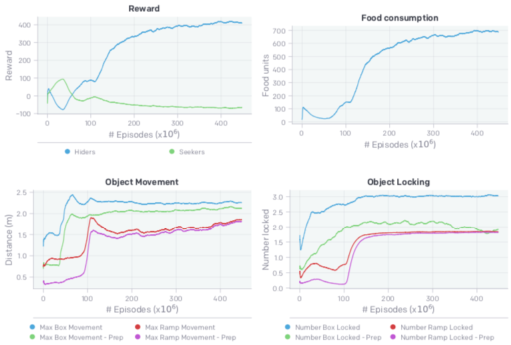


Figure: Hide and Seek with food reward results

Alternative games

Hide and Seek with dynamic food.

- Only one food that disappear when eaten.
- It reappears in the center of the map (in a square of length $1/5$ of game area size).
- The hidders need to learn to build large forts.
- emerge after 45 billions samples.
- if food region ratio is $1/6$, emerge after 15 billions samples.
- if food region ratio is $1/4$, the hiders ignore food and protect themselves.

Conclusions

- Simple game rules and multi-agent competition can induce agents to learn complex strategies and skills.
- Intelligent tests, using transfer learning, can be used to evaluate learning progress and to compare agents in a same domain.
- Multi-agent autocurricula can lead to physically grounded and human-relevant behavior (in opposition to unsupervised exploration techniques).

Future work

- Reduce the sample complexity: better policy learning algorithms, architecture and also reward functions.
- Generate environment without unwanted behaviors.

Surprising behaviors

See the blog.

References

- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. ,
- Schulman, John ; Wolski, Filip ; Dhariwal, Prafulla ; Radford, Alec ; Klimov, Oleg. (2017). Proximal Policy Optimization Algorithms.
- John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, Pieter Abbeel (2015) High-Dimensional Continuous Control Using Generalized Advantage Estimation
- Carlos Diuk, Andre Cohen, and Michael L. Littman. 2008. An object-oriented representation for efficient reinforcement learning.

References

- https://towardsdatascience.com/policy-gradients-in-a-nutshell-8b72f9743c5d
- https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f
- https://towardsdatascience.com/proximal-policy-optimization-ppo-with-sonic-the-hedgehog-2-and-3-c9c21dbed5e
- https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

The End