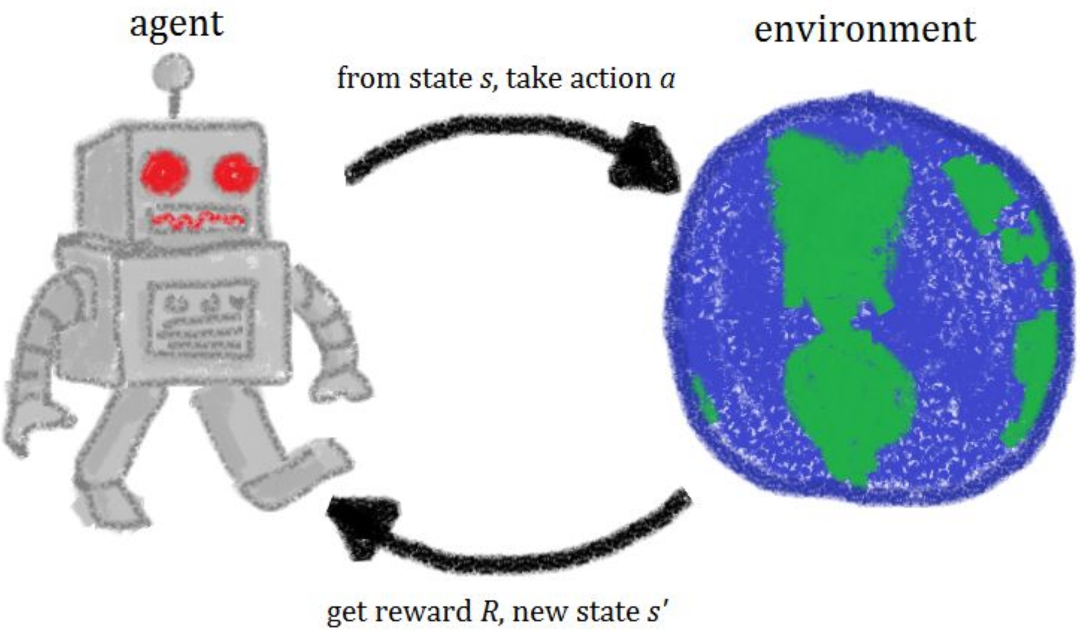
Intro to Reinforcement Learning

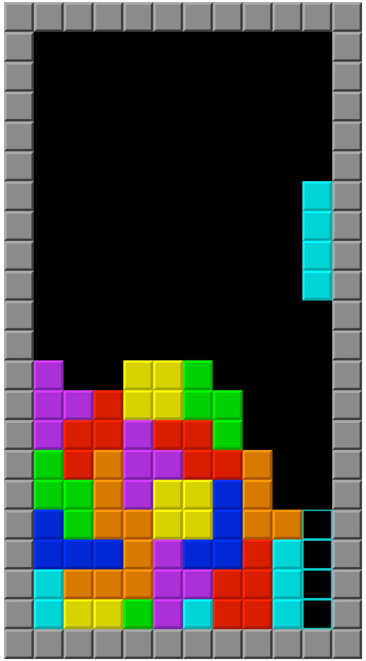
Markovich Anna

Who is this agent who uses reinforcement to beat the environment?

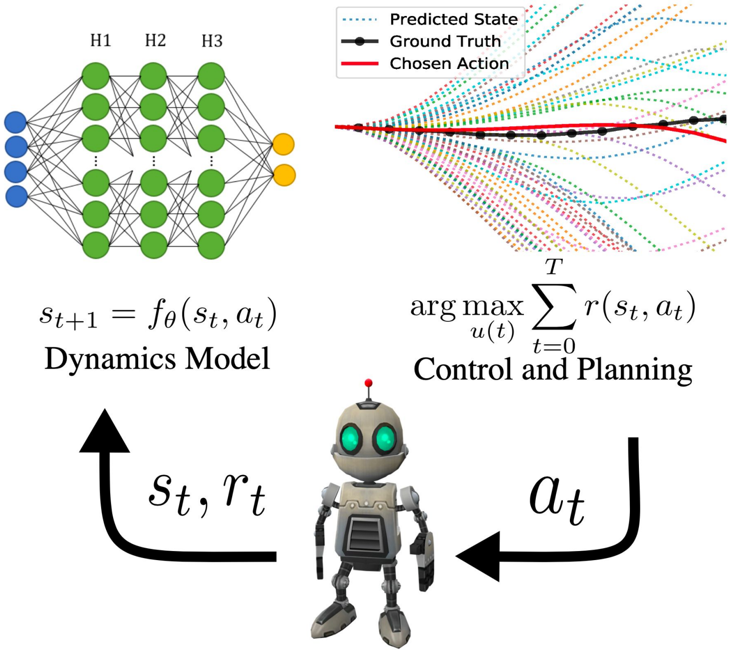


Environment | State | Action | Reward

Game: tetris



Neural network



RecSys

YouTube

Problem: how to learn optimal behaviour?

1. Exploration

2. Optimization

3. Generalization

- handle delayed consequences

Let's compare:

	Reinforcement learning	**Supervised learning**
Exploration	trial and error	None (known ground truth)
Optimization	sparse reward function	differentiable loss function
Generalization	update decision policy	avoid overfitting
Delayed consequences	reward may come afterwards	None

Formalism: Markov Decision Process

Markov assumption:

$$P(s_{t+1} | s_t, s_{t-1}, s_{t-2} \ldots s_1) = P(s_{t+1} | s_t)$$

(next state depends only on the current state)

Formalism: Markov Decision Process

Markov chain / Markov process – a stochastic model describing a sequence of possible events which satisfies Markov assumption.

(S, P):

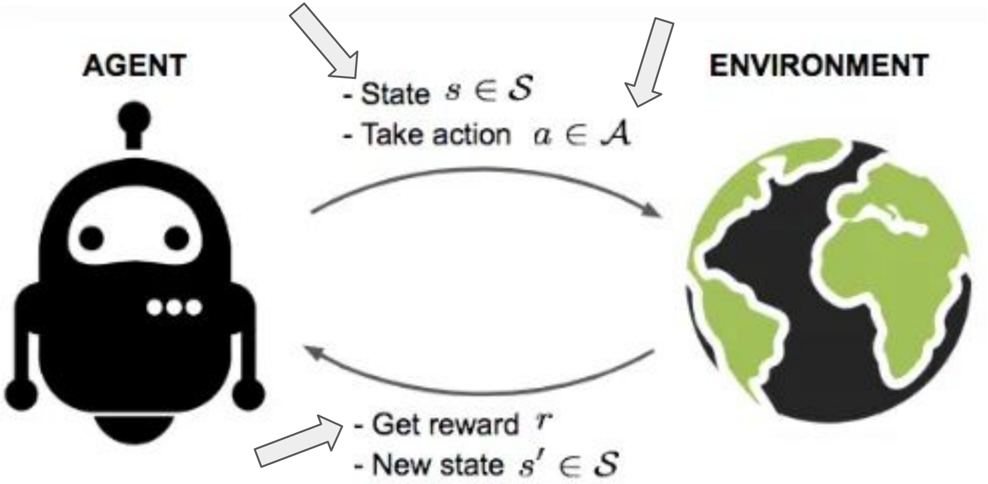
S – state space $\qquad s \in S$

P– dynamics (conditional distribution) $\qquad P(s', s) = P(s_{t+1} = s' | s_t = s)$

Formalism: Markov Decision Process

MDP = Markov chain + action + reward + policy

$$P(s', s, a) = P(s_{t+1} = s' | s_t = s, a_t = a) \qquad \pi(a|s) = P(a_t = a | s_t = s)$$



$$R(s, a) = \mathbb{E}[r | s_t = s, a_t = a]$$

Total reward / Return

total rewards for session

$$G_t = \sum_{s=t}^{T} R_s$$

with discount

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots \gamma^T R_{t+T}$$

$$\gamma \in (0, 1)$$

Goal

find policy :

$$\mathbb{E}_\pi[G_t] \to \max_\pi$$

Cross-entropy method

0. initialize policy

loop:

1. sample N sessions
2. choose $M < N$ **elite** sessions
3. update policy to favor pairs (a, s) from **elite**

Cross-entropy method

PyTorch Tabular

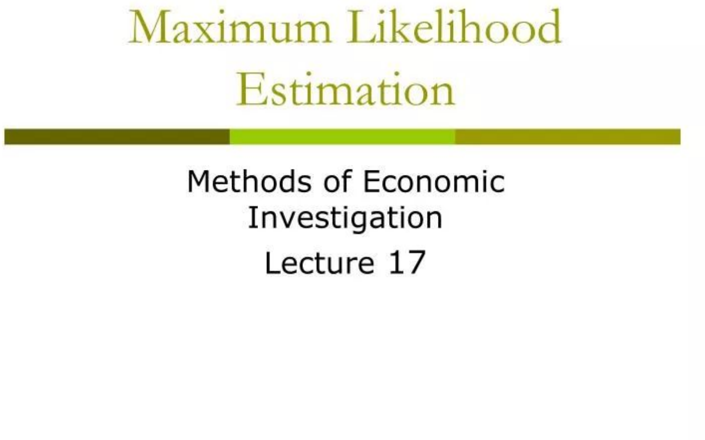
Tabular : finite state & action spaces

policy matrix: $\pi(a|s) = \Pi_{a,s} \in \mathbb{R}^{|A|\times|S|}$

update rule: $$\pi(a_p|s_k) = \frac{\sum_{(a,s)\in elite} [s_k = s][a_p = a]}{\sum_{(a,s)\in elite} [s_k = s]}$$

Cross-entropy method

Infinite/large state space

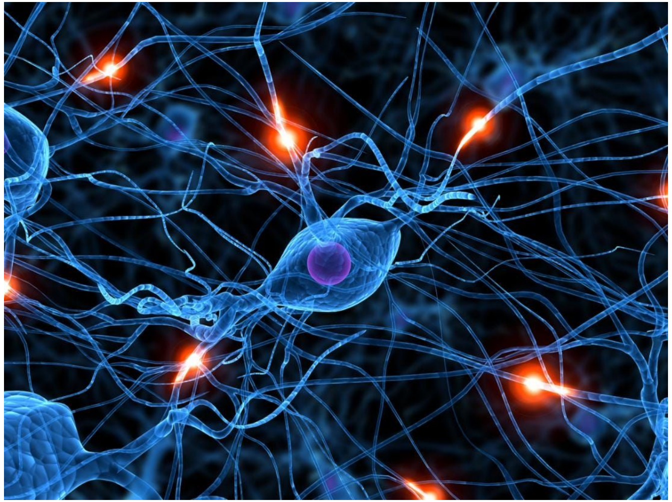


loop:

1. sample N sessions
2. choose M elite sessions
3. $\pi = \underset{\pi}{argmax} \sum_{(a_i,s_i)\in elite} \log(\pi(a_i|s_i))$

Cross-entropy method

Infinite/large state space



Using NNs:

0. net = MLPClassifier(...)

loop:

1. sample N sessions
2. choose M elite sessions
3. net.fit(elite_states, elite_actions)*

$$^{*}W_k = W_{k-1} + \eta_k \nabla\Big(\sum_{(a_i,s_i)\in elite} \log(\pi_{W_{k-1}}(a_i|s_i))$$

Cross-entropy method

Continuous state & action space

model: $\pi(a|s) \sim \mathcal{N}(\mu(s), \sigma^2)$

μ - NN output

σ - parameter / other NN output

Using NNs:

0. net = MLPRegressor(...)

loop:

1. sample N sessions
2. choose M elite sessions
3. net.fit(elite_states, elite_actions)*

$$*W_k = W_{k-1} + \eta_k \nabla\Big(\sum_{(a_i,s_i)\in elite} \log(\pi_{W_{k-1}}(a_i|s_i))$$

Intrigue: better method for optimal policy search

(not today)

Remember ?

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots \gamma^T R_{t+T}$$

$$\mathbb{E}_\pi[G_t] \to \max_\pi$$

V-function and Q-function

State-value function

$$V_\pi(s) = \mathbb{E}_\pi[G_t | s_t = s]$$

expected return
conditional on state

Action-value function

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t | s_t = s, a_t = a]$$

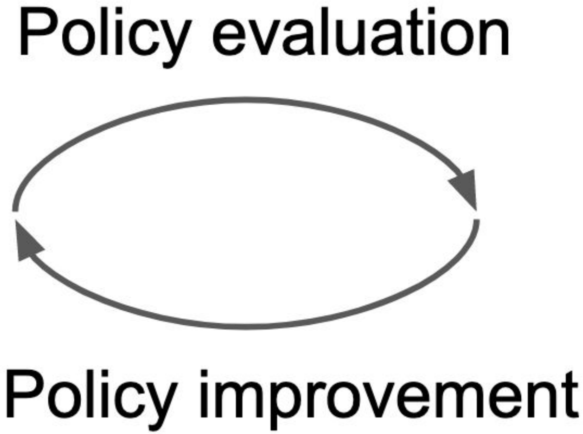
expected return
conditional on state and action

Complex mathematical calculations

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{r,s'} p(r, s'|s, a)[r + \gamma V_\pi(s')] = \sum_a \pi(a|s) Q_\pi(s, a)$$

$$Q_\pi(s, a) = \sum_{r,s'} p(r, s'|s, a)[r + \gamma V_\pi(s')]$$

Policy iteration, Value iteration



$\Rightarrow$

1. deterministic policy :

$$\pi : S \to A$$

2. $\pi(s) \leftarrow \underset{a}{\operatorname{argmax}} \quad Q_\pi(s, a)$

Sources

https://github.com/yandexdataschool/Practical_RL

http://web.stanford.edu/class/cs234/