Lecture 07: Intro to Reinforcement Learning

Radoslav Neychev

Fall 2020, Moscow, Russia

Outline

1. Reinforcement Learning problem statement
2. (Multi-armed) bandits
3. MDP formalism
4. Relations to Psychology
5. Cross-entropy method
6. Reinforcement, Supervised and Unsupervised Learning

Based on: YSDA Practical RL course, Berkeley CS188 Intro to AI. Several icons by [Smashicons]

Reinforcement Learning

problem statement

Supervised learning

- Given:
  - Objects and reference answers $\boxed{x \in \mathcal{X}, y \in \mathcal{Y}}$ Want them to be i.i.d.
  - Loss/objective function $\boxed{L(\hat{y}, y)}$ Usually differentiable
  - Model family $f \in \mathcal{F}, f : \mathcal{X} \longrightarrow \mathcal{Y}$
- Goal:
  - Find optimal mapping $f^* = \arg\min_f L(f(x), y)$

Reinforcement learning

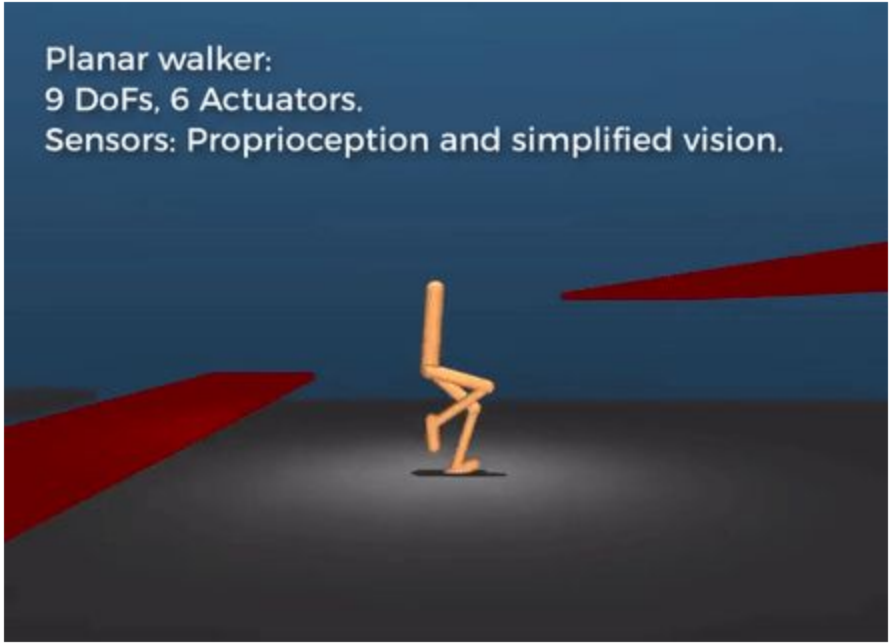
- Given:
  - Objects ~~and reference answers~~ $x \in \mathcal{X}$
  - Loss/objective function $\boxed{L(\hat{y}, y)}$
  - Model family $f \in \mathcal{F}, f : \mathcal{X} \longrightarrow \mathcal{Y}$

Usually no reference answers

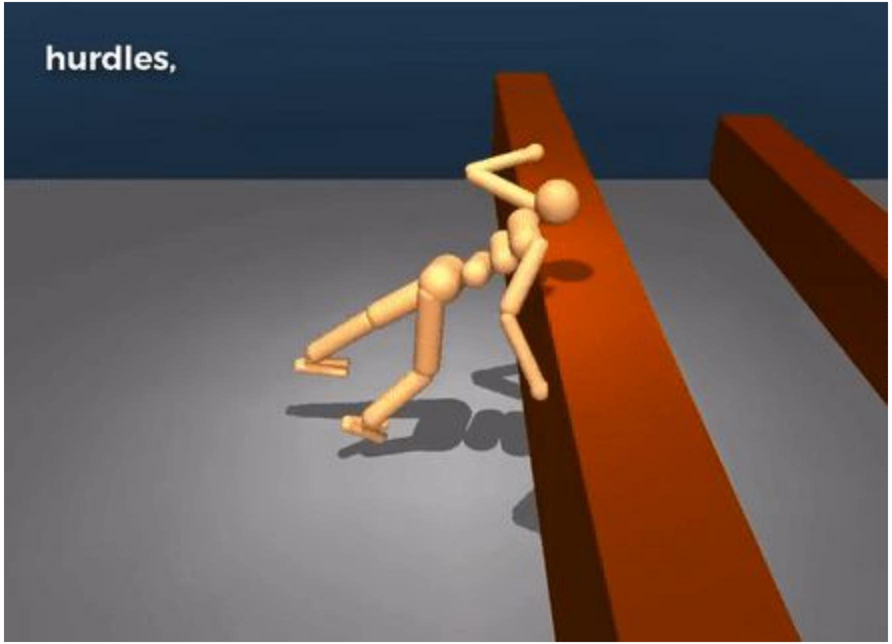
E.g. want the robot to walk

Usually even hard to formulate, non-differentiable

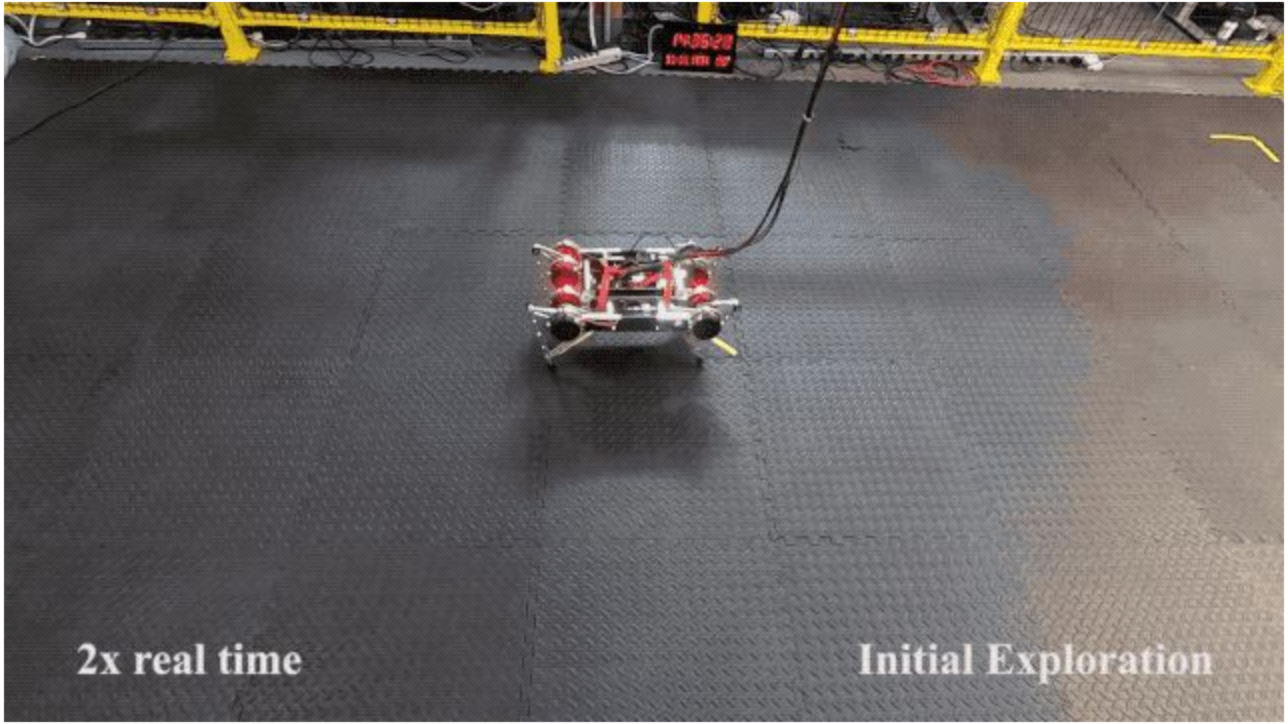
- Goal:
  - Find optimal mapping $f^* = \arg\min_f L(f(x), y)$



source: Emergence of Locomotion Behaviours in Rich Environments, DeepMind

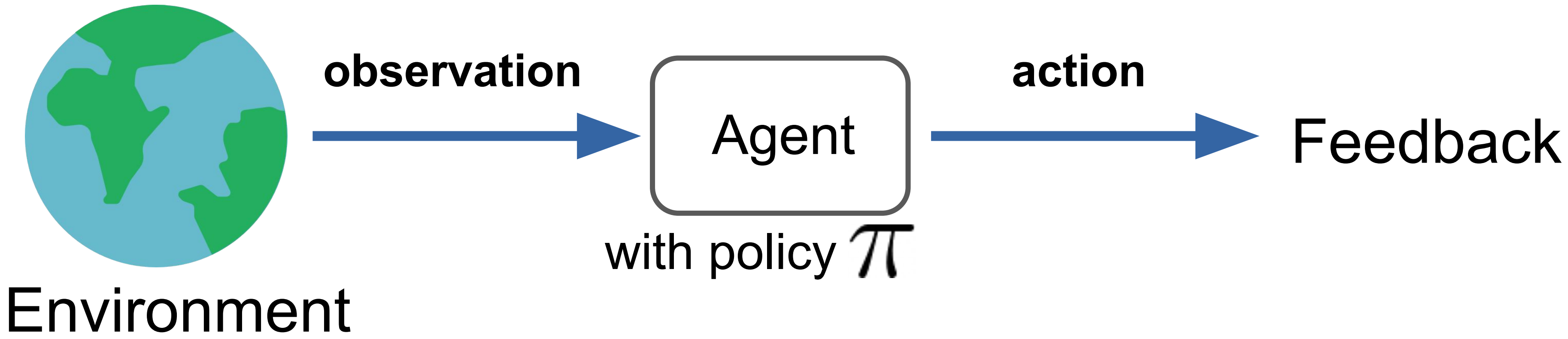


source: Emergence of Locomotion Behaviours in Rich Environments, DeepMind

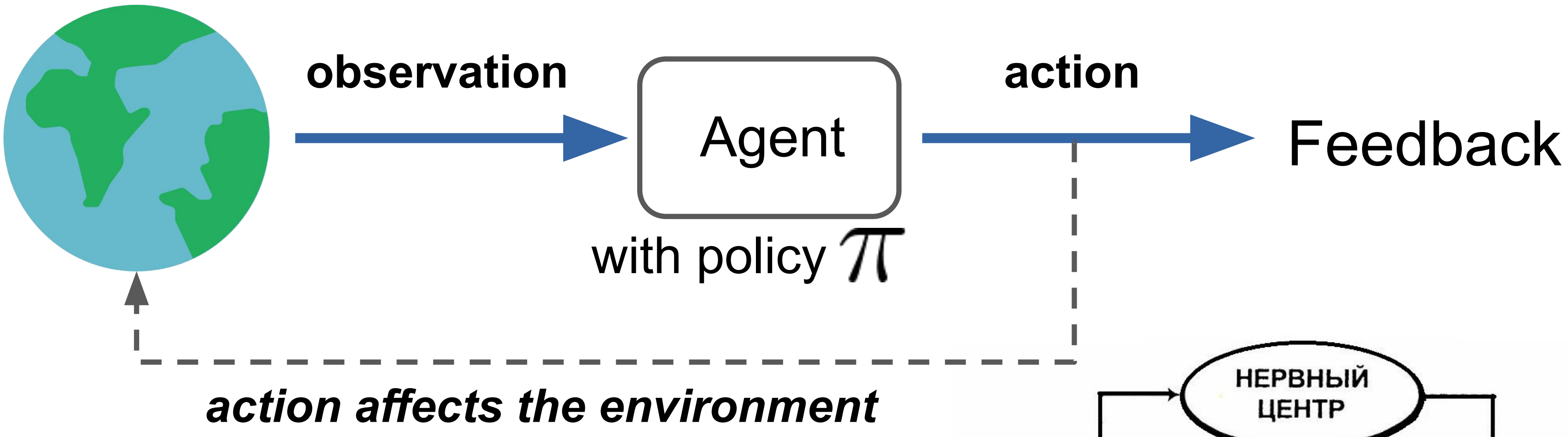


source: [MIT Technology Review]

(Multi-armed) bandits

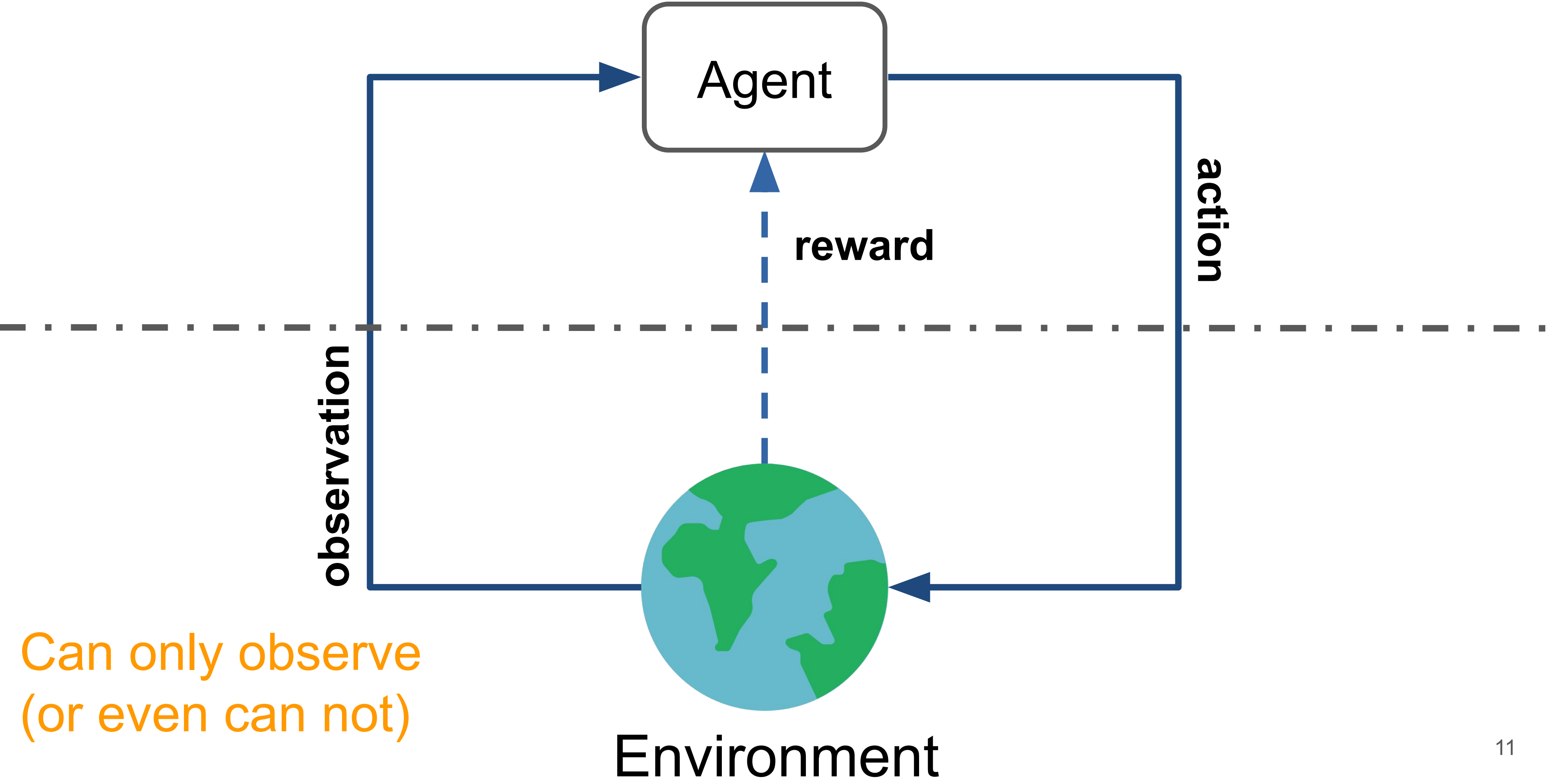


- Observation (state): vector or image or sequence … or *nothing*
- Policy: mapping from state to action
- Action
- Feedback (reward): usually a converted to a number



Рефлекторное кольцо введено А. Ф. Самойловым (ученик И.П. Павлова) в 1930, также исследовалось Н.А. Бернштейном

We are in control



Can only observe
(or even can not)

Reality check: dynamic control

Variety of papers on helicopter control:
heli.stanford.edu

Andrew Y. Ng PhD Thesis link: "Shaping and policy search in Reinforcement Learning"

- Observation: accelerometer, gyroscope, engine data
- Action: change rotation speed, angle
- Feedback: some specific reward

source: heli.stanford.edu, photos by Ben Tse and Eugene Fratkin

~~Reality~~ check: video games

- Observation: image(s)
- Action: move, fire, turn
- Feedback: score/health/progres/…

source: gym.openai.com, DOOM

Open questions

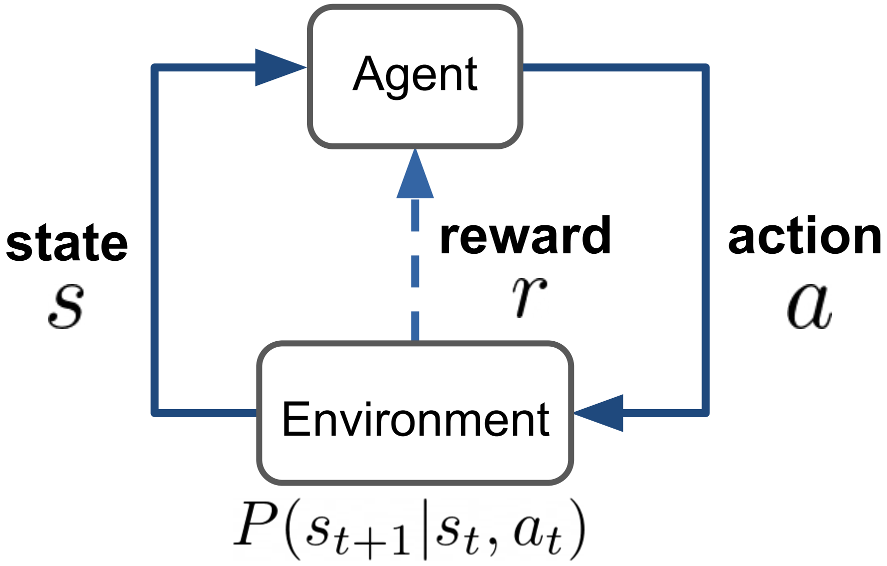
- What is *optimal* action?
  - Maximize the reward **on the next step**
  - Maximize the reward **in long term**

Open questions

- *Explore* or *exploit*?
  - Stepping of *current optimal* strategy may **decrease** the cumulative reward
  - Under *current optimal strategy* one may **never discover** something better

MDP formalism

- State: $s \in \mathcal{S}$
- Action: $a \in \mathcal{A}$
- Reward: $r \in \mathbb{R}$



- Dynamics: $P(s_{t+1}|s_t, a_t)$

Markov property:

$$P(s_{t+1}|s_t, a_t, \ldots, s_0, t_0) = P(s_{t+1}|s_t, a_t)$$

Total reward

- Total reward for session: $R = \sum_t r_t$

- Policy: $\pi(a|s) = P(\text{take action } a \text{ in state } s)$

- Goal: maximize reward; $\pi^*(a|s) = \arg\max_\pi \mathbb{E}_\pi[R]$

Psychological point of view

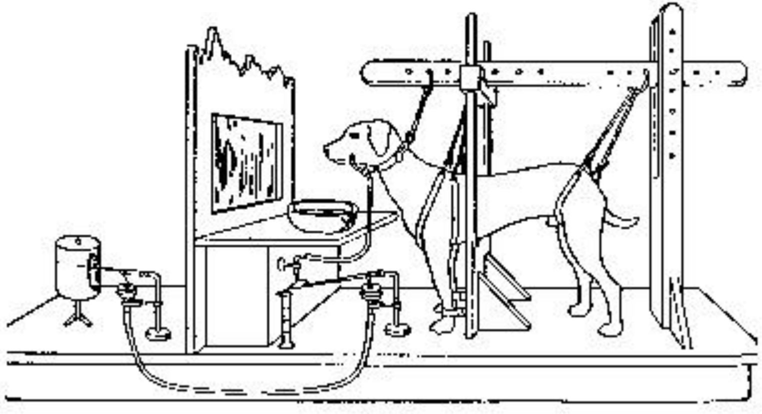


Fig. 2.

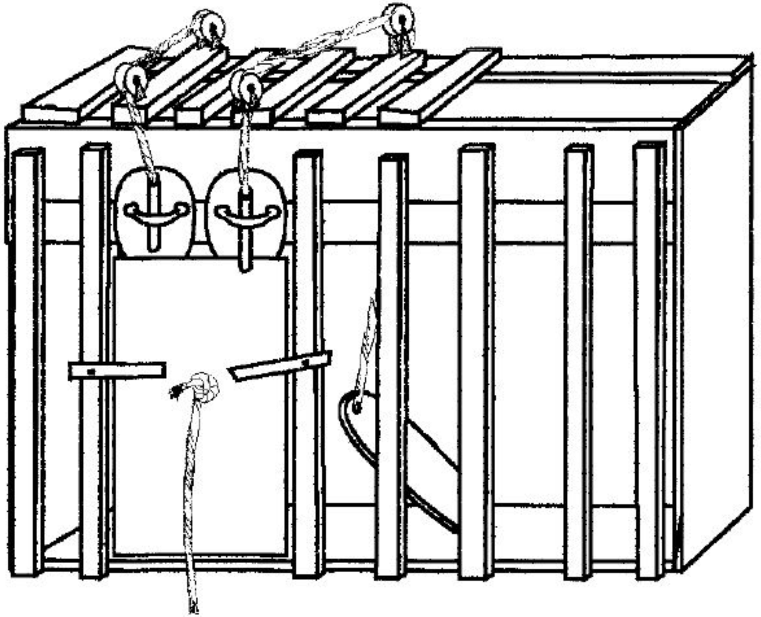
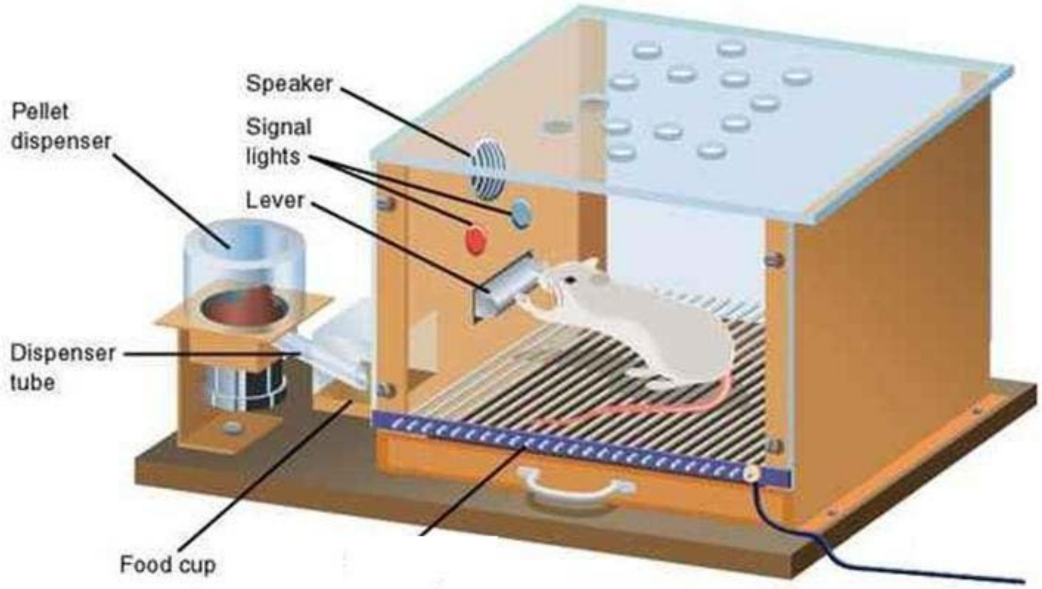


Fig. 4. Box K. The door is held in place by a weight suspended by a string. To open the door, a cat had to depress a treadle, pull on a string, and push a bar up or down. (After Thorndike, 1898, Figure 1, p. 8.)

source: Thorndike experiment: [original paper (1898)] and [blog post]
[Pavlov's Classical conditioning on Wikipedia]

Psychological point of view



source: [Operant conditioning chamber (Skinner box) on Wikipedia](), [Biomechatronic design and development of a legged rat robot]() paper

How to maximize the reward?

$\mathbb{E}_\pi[R]$ is an expected cumulative reward earned per session following policy π

Need to maximize the following objective:

$$\mathbb{E}_\pi[R] = \mathbb{E}_{s_0 \sim P(s_0)} \mathbb{E}_{a_0 \sim \pi(a|s_0)} \cdots \mathbb{E}_{s_t, r_t \sim P(s, r|s_{t-1}, a_{t-1})} [r_0 + \ldots + r_t]$$

How to do it?

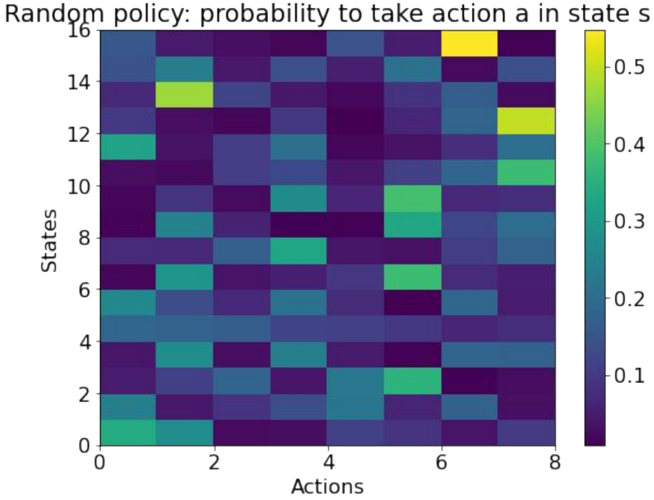
How to maximize the reward?

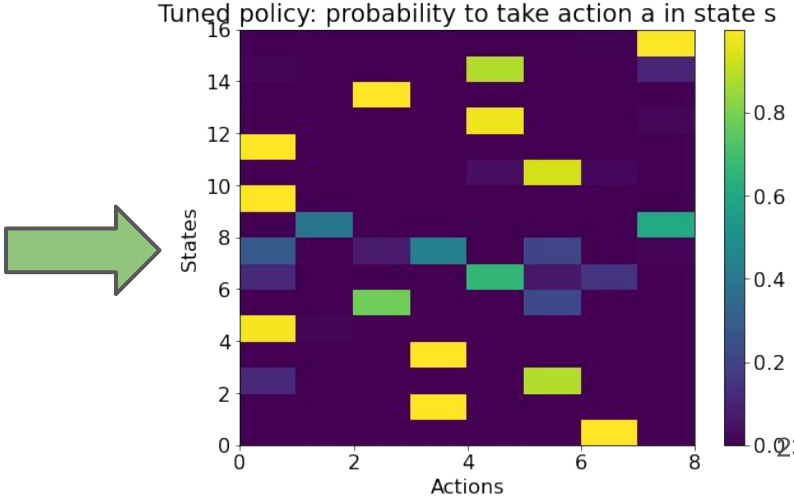
- Play a few sessions with existing policy
- Update the policy using new feedback
- Repeat

Cross-entropy method

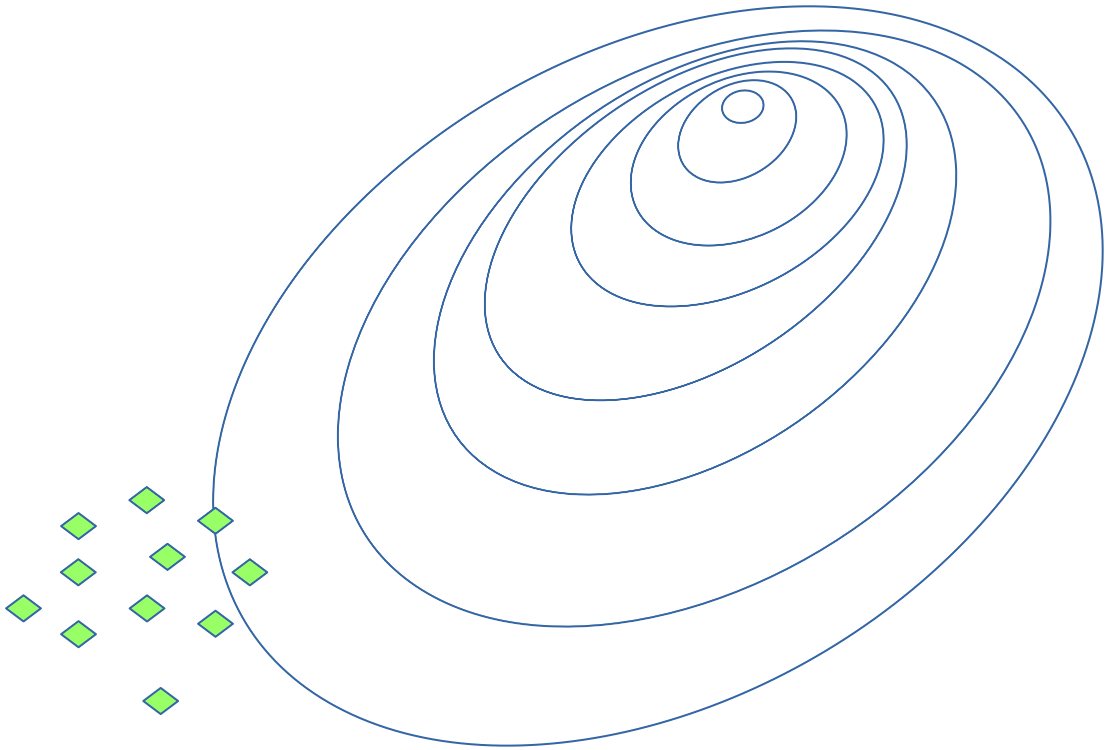
Cross-entropy method: tabular case

- Initialize policy (state-action matrix, every row sums up to 1)
- Sample N sessions
- Select M **elite** sessions with highest rewards
- Update policy using the elite session state-action sequences
- Repeat

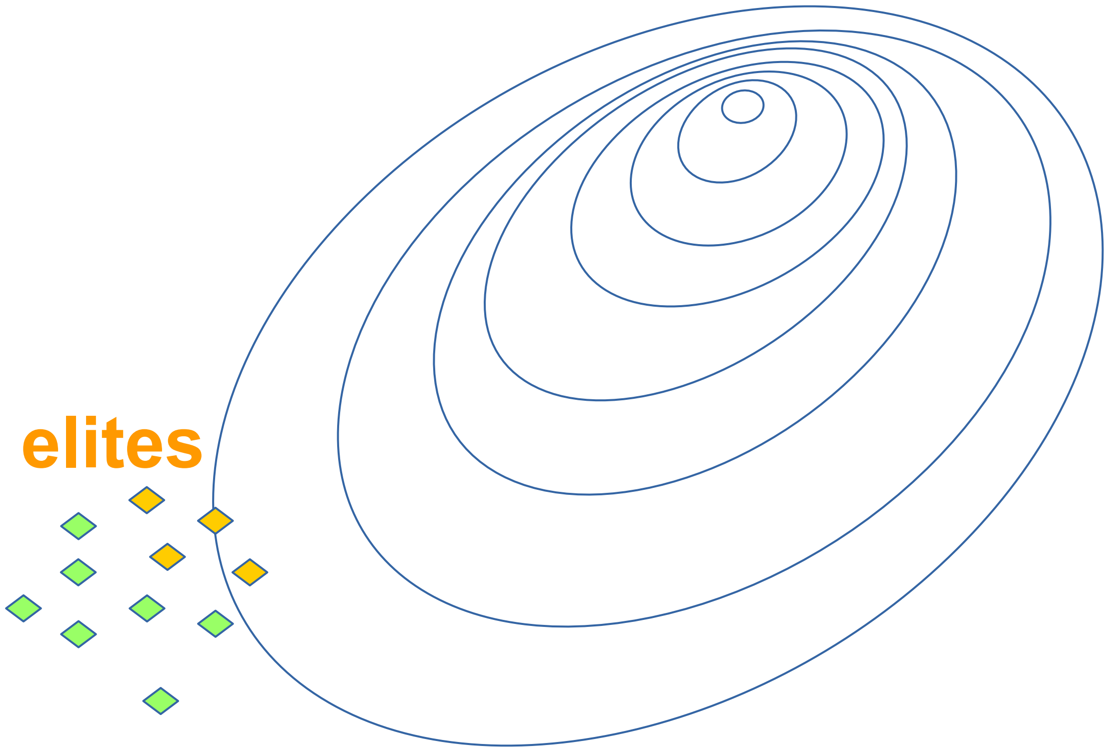




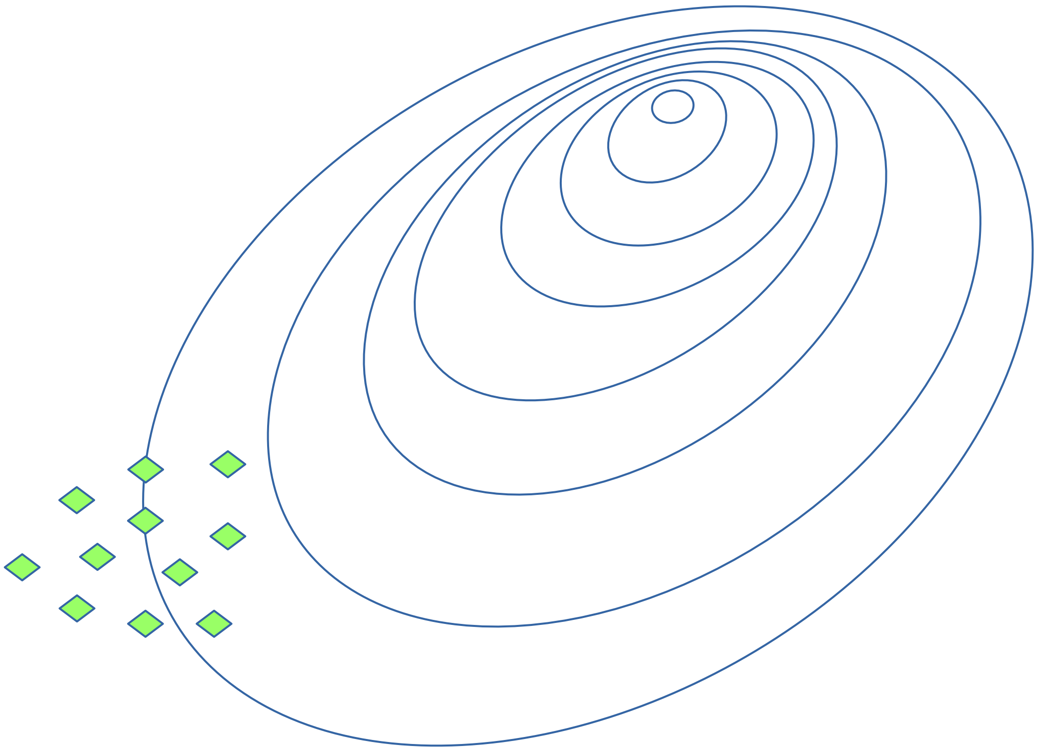
Cross-entropy method: illustration



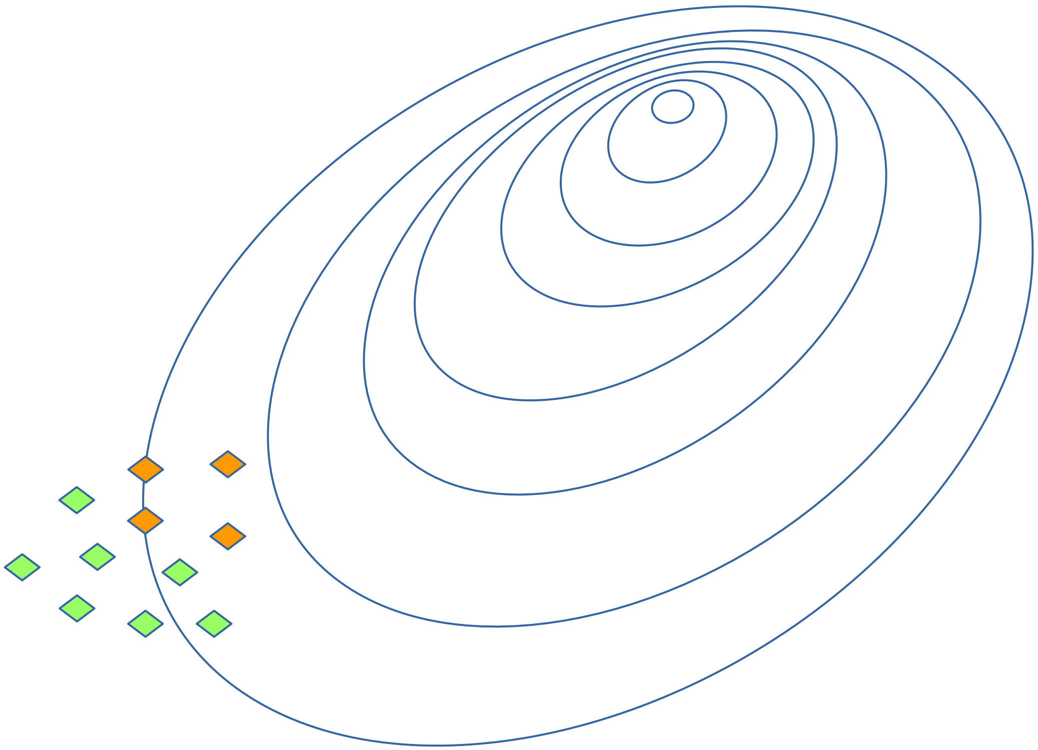
Cross-entropy method: illustration



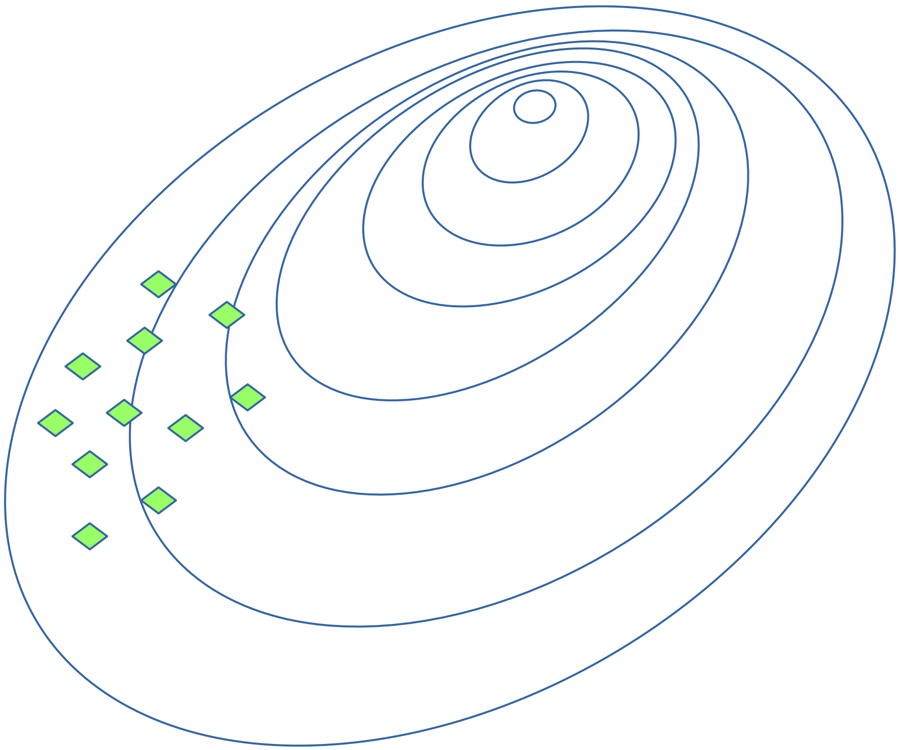
Cross-entropy method: illustration



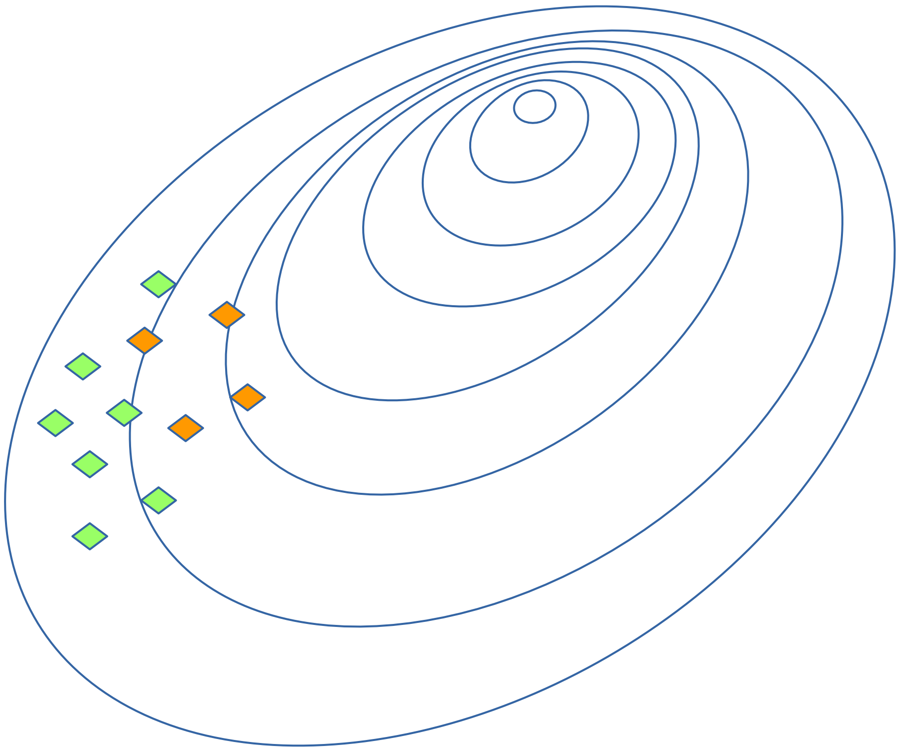
Cross-entropy method: illustration



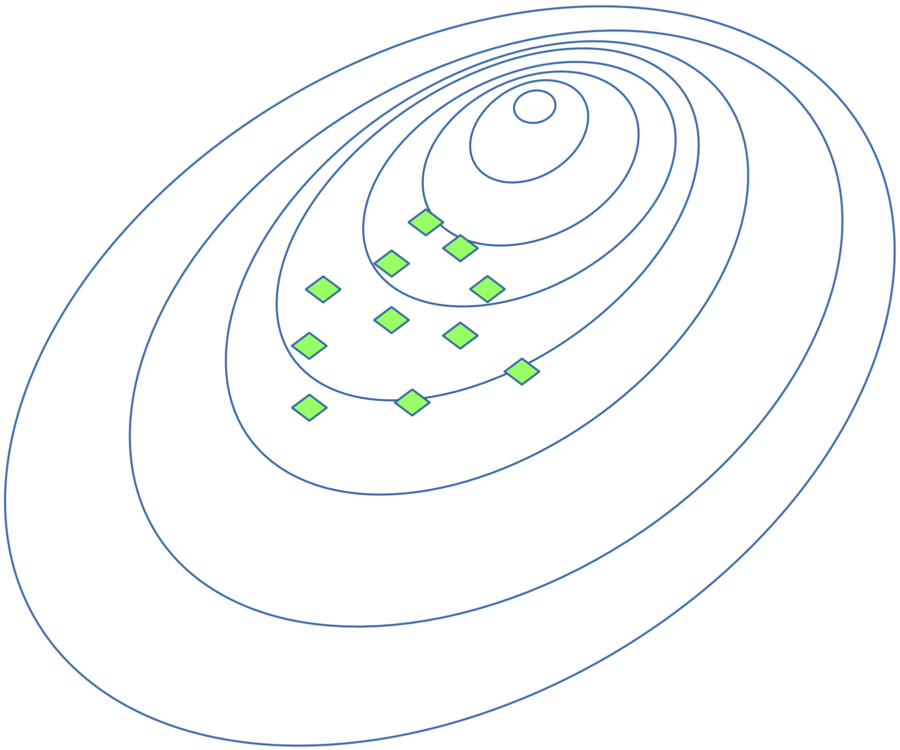
Cross-entropy method: illustration



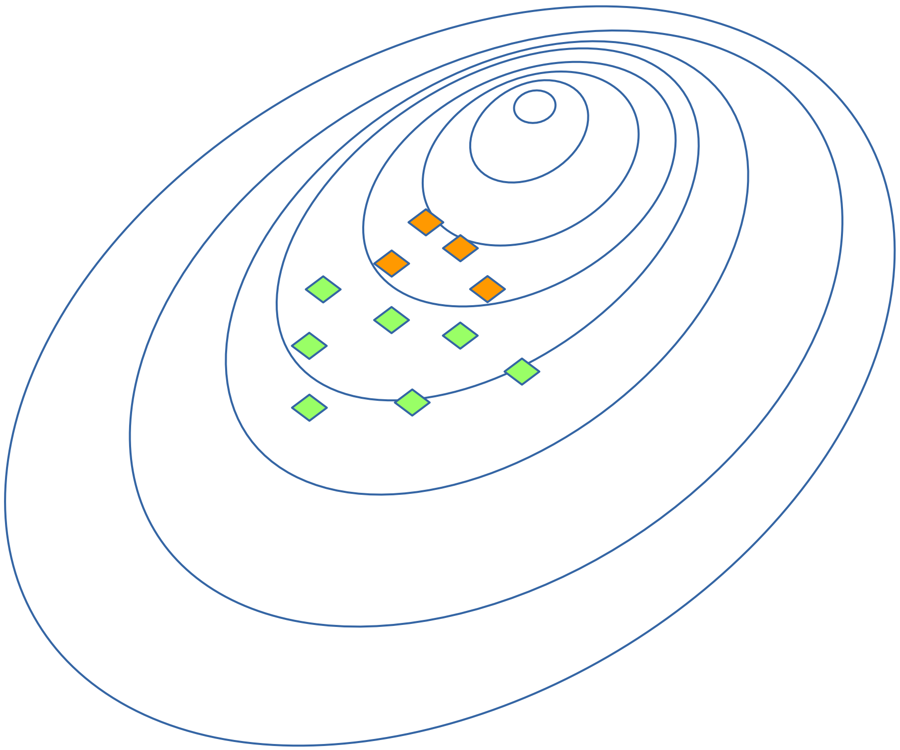
Cross-entropy method: illustration



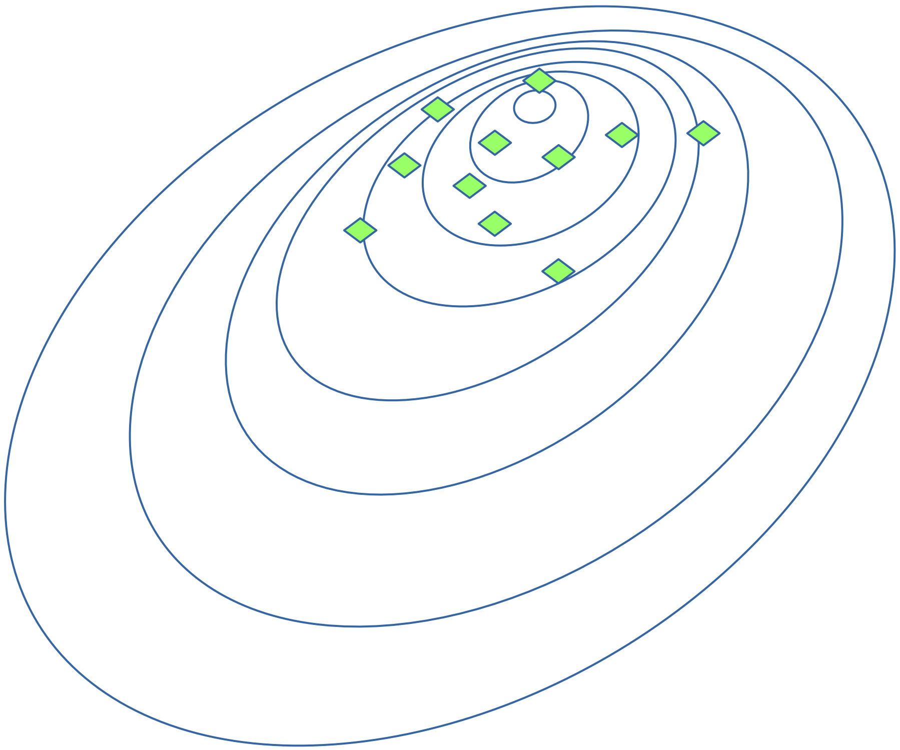
Cross-entropy method: illustration



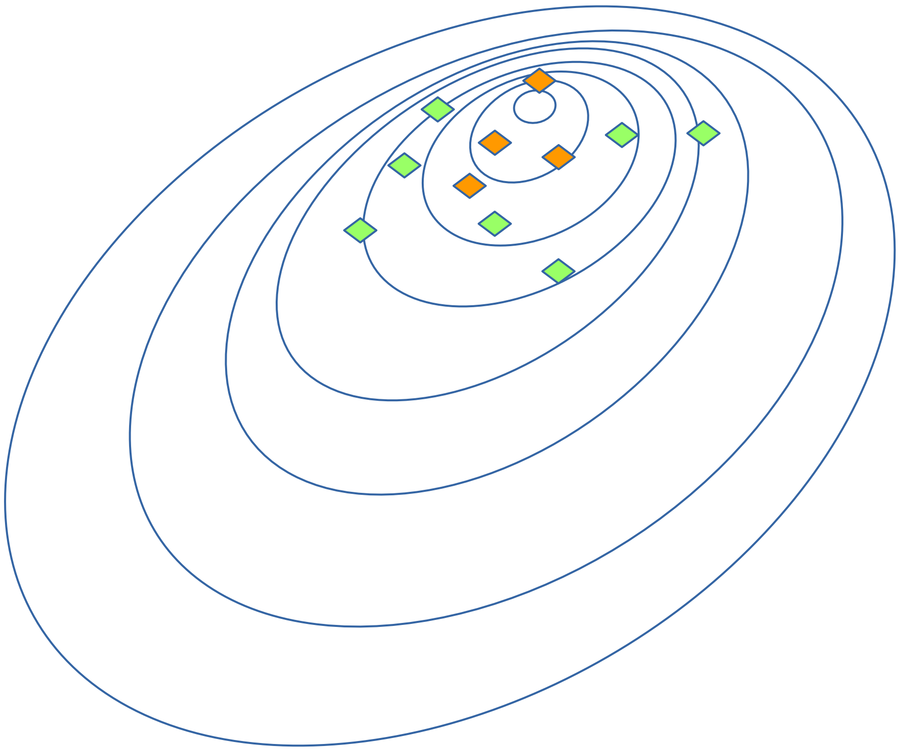
Cross-entropy method: illustration



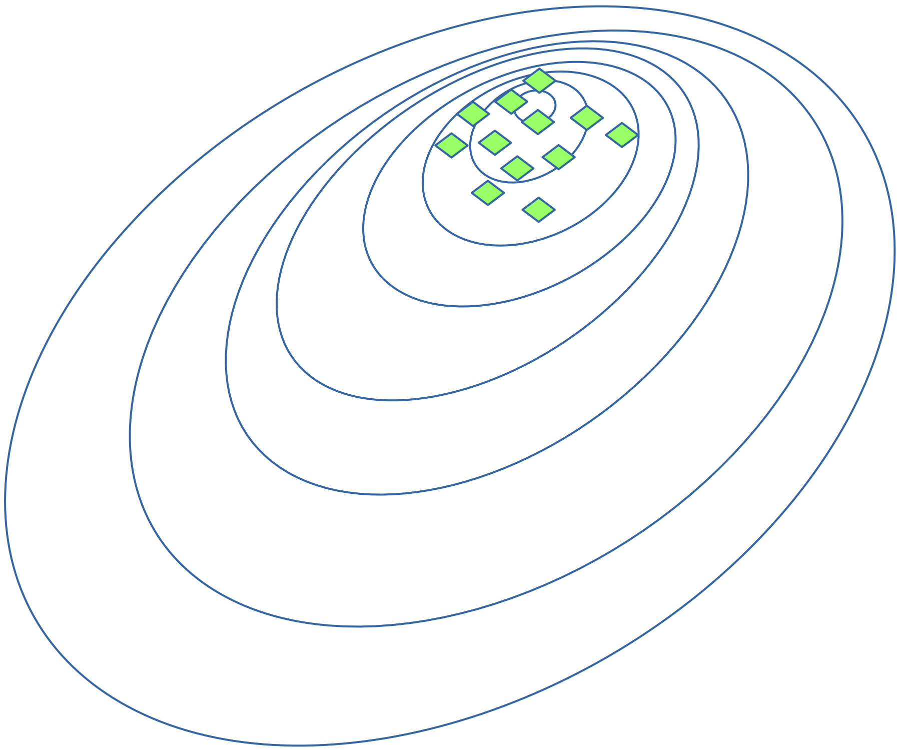
Cross-entropy method: illustration



Cross-entropy method: illustration



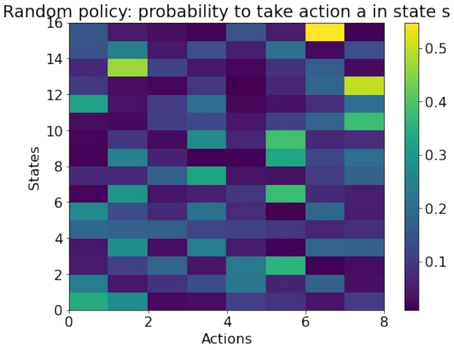
Cross-entropy method: illustration



Cross-entropy method: tabular case

- Policy is a matrix

$$\pi(a|s) = A_{s,a} \Longleftrightarrow$$



- Sample N games with this policy

- Select M **elite** sessions with highest rewards

$$\text{Elite} = [(s_0, a_0), (s_1, a_1), \ldots, (s_M, a_M)]$$

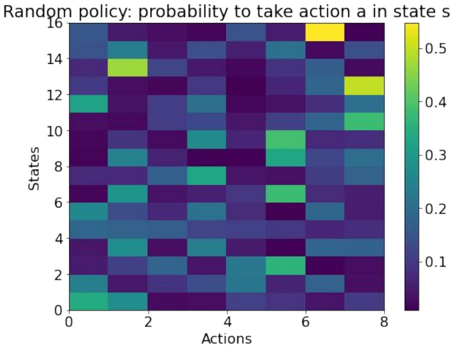
- Update policy:

$$\pi_{\text{new}}(a|s) = \frac{\sum\limits_{s_t, a_t \in \text{Elite}} [s_t = s][a_t = a]}{\sum\limits_{s_t, a_t \in \text{Elite}} [s_t = s]}$$

Cross-entropy method: tabular case

- Policy is a matrix

$$\pi(a|s) = A_{s,a}$$



- Sample N games with this policy
- Select M **elite** sessions with highest rewards
- Update policy using the **elite** sessions:

$$\pi_{\text{new}}(a|s) = \frac{\text{how many times took action } a \text{ at state } s}{\text{how many times was at state } s}$$

Harsh reality

Some environments have huge or infinite number of states

How to fix it?

Approximate cross-entropy method

- Model (e.g. parametric) predicts action probability given state:

$$\pi(a|s) = f_\theta(a, s)$$

```
model = RandomForestClassifier()
```

Random Forest Classifier, Logistic Regression, NN etc.

- Sample N sessions, select M **elite** sessions

$$\text{Elite} = [(s_0, a_0), (s_1, a_1), \ldots, (s_M, a_M)]$$

New training set; states are objects, actions are target values

- Maximize likelihood of actions in elite sessions:

$$\pi(a|s)_{\text{new}} = \arg\max_\pi \sum_{s_t, a_t \in \text{Elite}} \log \pi(a_i|s_i)$$

```
model.fit(elite_states, elite_actions)
```

What if action space is continuous?

Approximate cross-entropy method

- Model samples actions from some appropriate distribution:

$$\pi(a|s) = \mathcal{N}(\mu_\theta(a, s), \sigma_\gamma(a, s))$$

One model   Another model (or constant)

It is just a regressor!

Approximate cross-entropy method

What if action space is continuous?

- Model (e.g. parametric) predicts action given state:

```
model = RandomForestRegressor()
```

- Sample N sessions, select M **elite** sessions

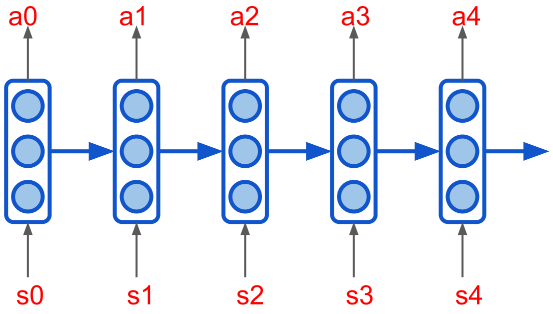
$$\text{Elite} = [(s_0, a_0), (s_1, a_1), \ldots, (s_M, a_M)]$$

- Maximize likelihood of actions in elite sessions:

```
model.fit(elite_states, elite_actions)
```

Useful ideas

- Use elite sessions from several (3-5) past iterations for training
    - Experience from previous iterations is preserved
    - Convergence may be slower (e.g. on simple environments)
- Regularize the policy with entropy
    - Low entropy means weak exploration
- Sessions can be sampled in parallel
- Agent can use memory as well
    - We will meet RNNs again soon



Key differences

Key differences

Supervised Learning

- Learn to approximate reference answers
- Need reference answers
- Model does not affect the input data

Reinforcement Learning

- Learn optimal strategy by trial and error
- Need feedback on agent’s actions
- Agent actions affect the environment (so the observations)

Key differences

Unsupervised Learning

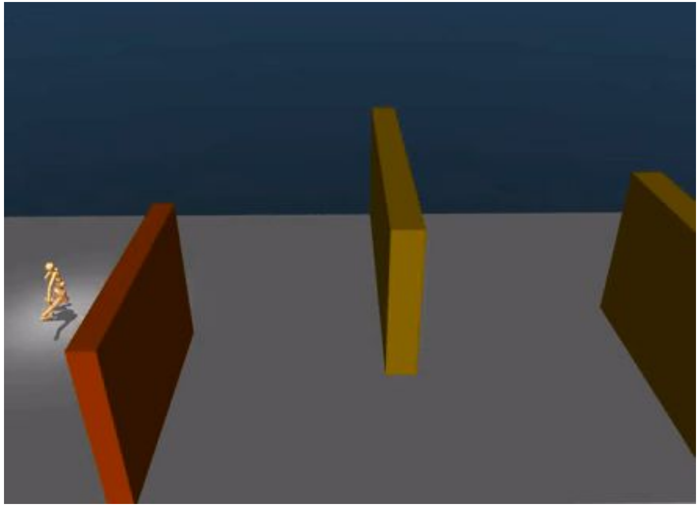
- Learn underlying data structure
- No feedback required
- Model does not affect the input data

Reinforcement Learning

- Learn optimal strategy by trial and error
- Need feedback on agent’s actions
- Agent actions affect the environment (so the observations)

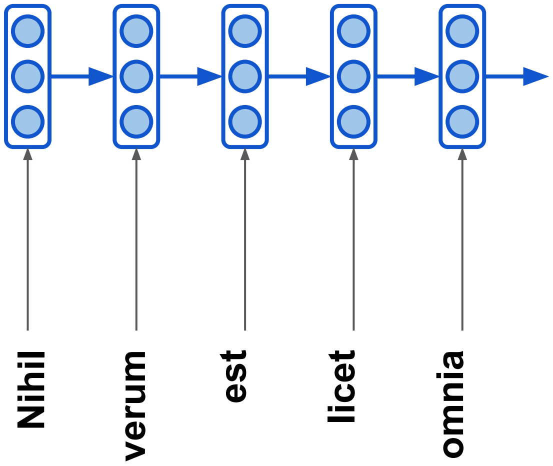
Outro

- RL is different both from Supervised and Unsupervised learning
- Reward formulation has huge effect on the agent behaviour
- Remember the Markov assumptions
- The crossentropy method is simple and still very powerful approach

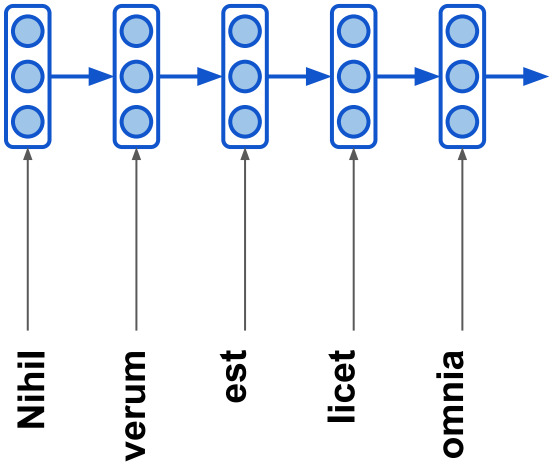


source: [Emergence of Locomotion Behaviours in Rich Environments](), DeepMind

MDP formalism



RNNs and dropout



Duct tape approach