Learning Action Representations for Reinforcement Learning

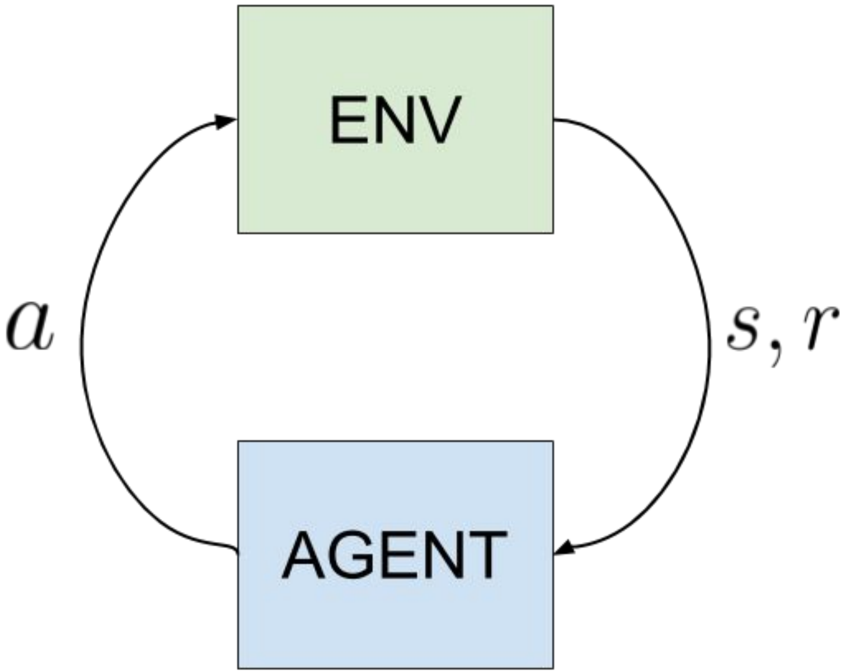




Yash Chandak
Georgios Theocharous
James Kostas
Scott Jordan
Philip Thomas

Reinforcement Learning

$$s = \text{State}$$
$$a = \text{Action}$$
$$r = \text{Reward}$$



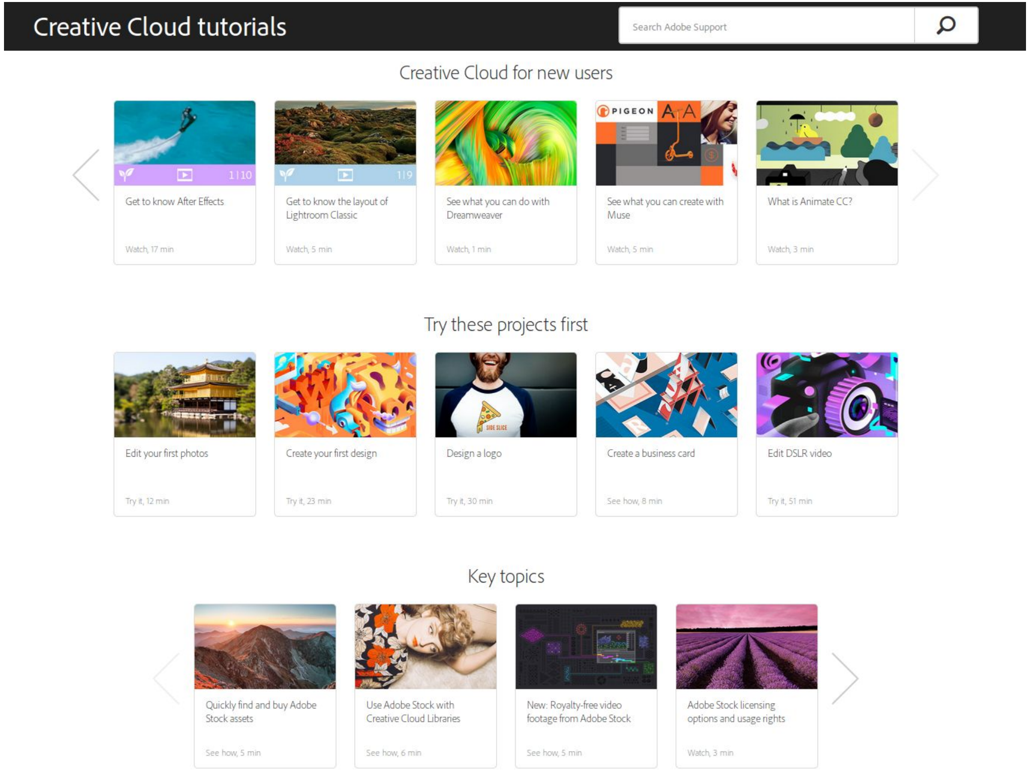
Problem Statement

Thousands of possible actions!

Problem Statement

Thousands of possible actions!

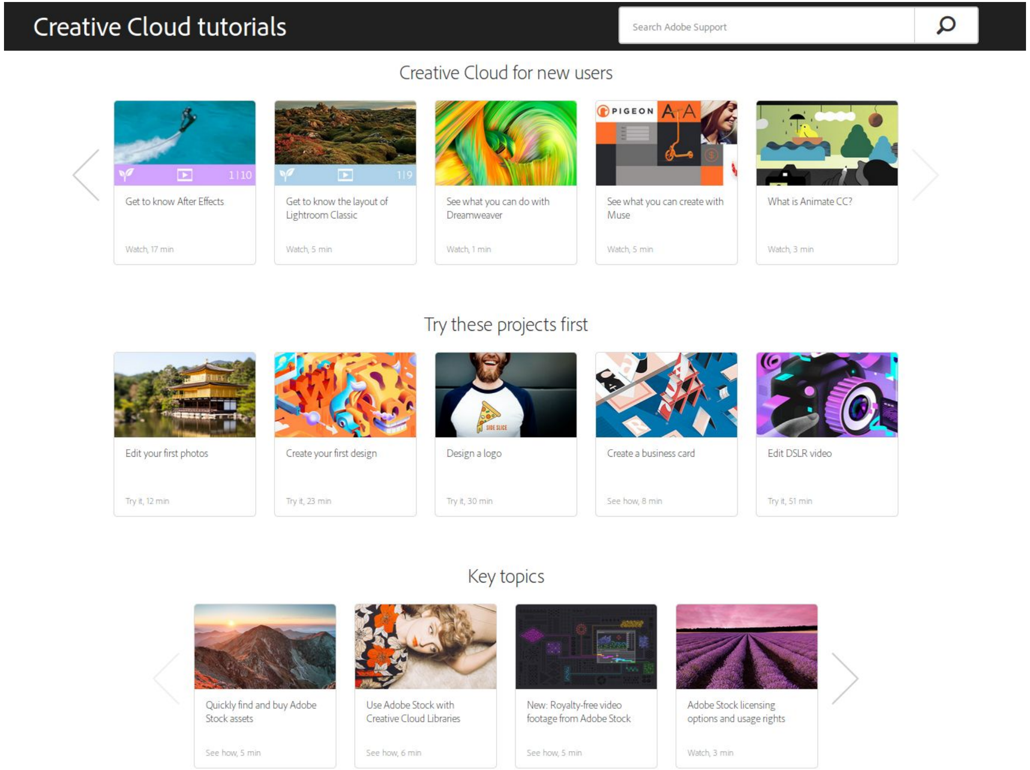
- Personalized tutoring systems



Problem Statement

Thousands of possible actions!

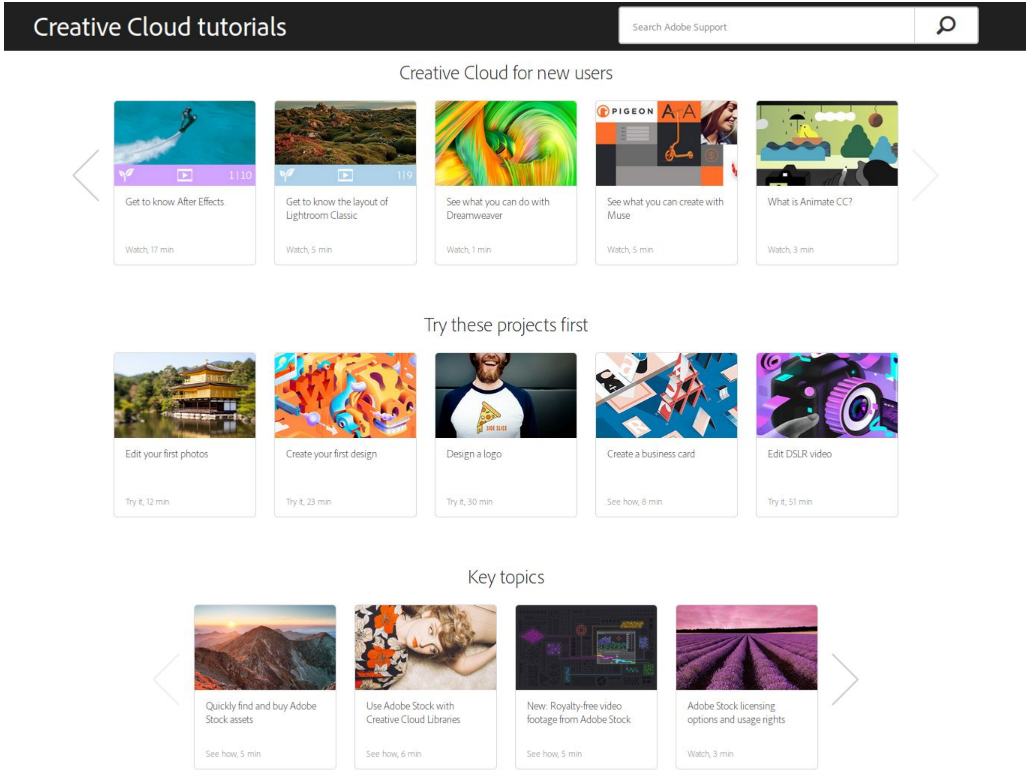
- Personalized tutoring systems
- Advertisement/marketing



Problem Statement

Thousands of possible actions!

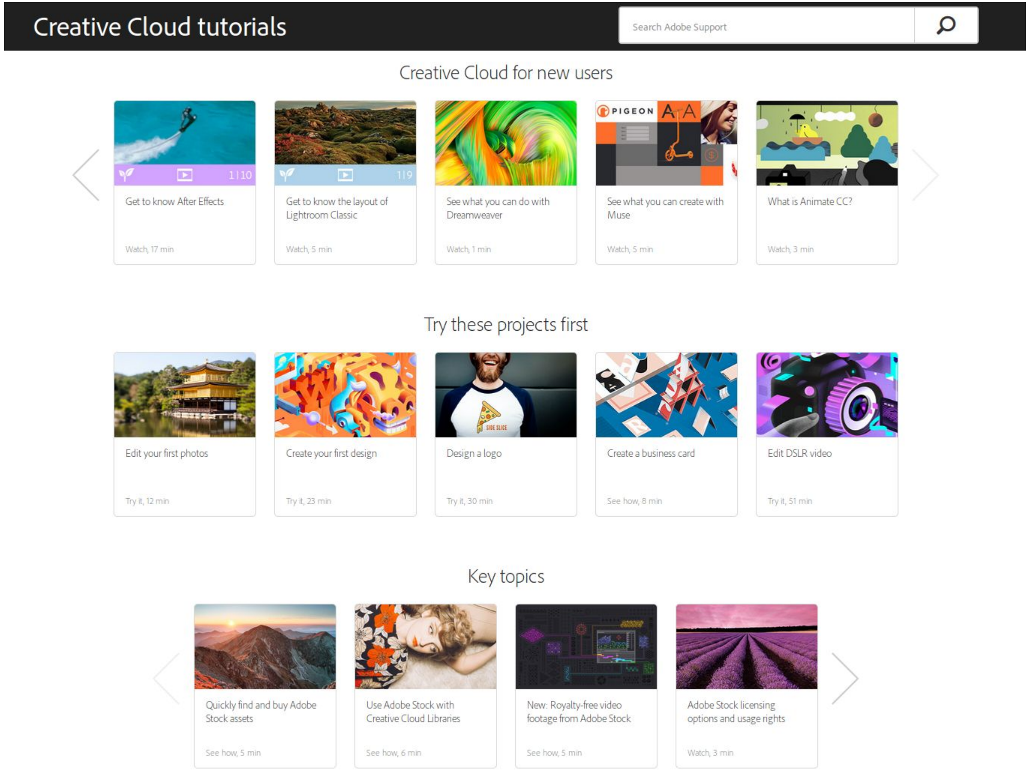
- Personalized tutoring systems
- Advertisement/marketing
- Medical treatment - drug prescription



Problem Statement

Thousands of possible actions!

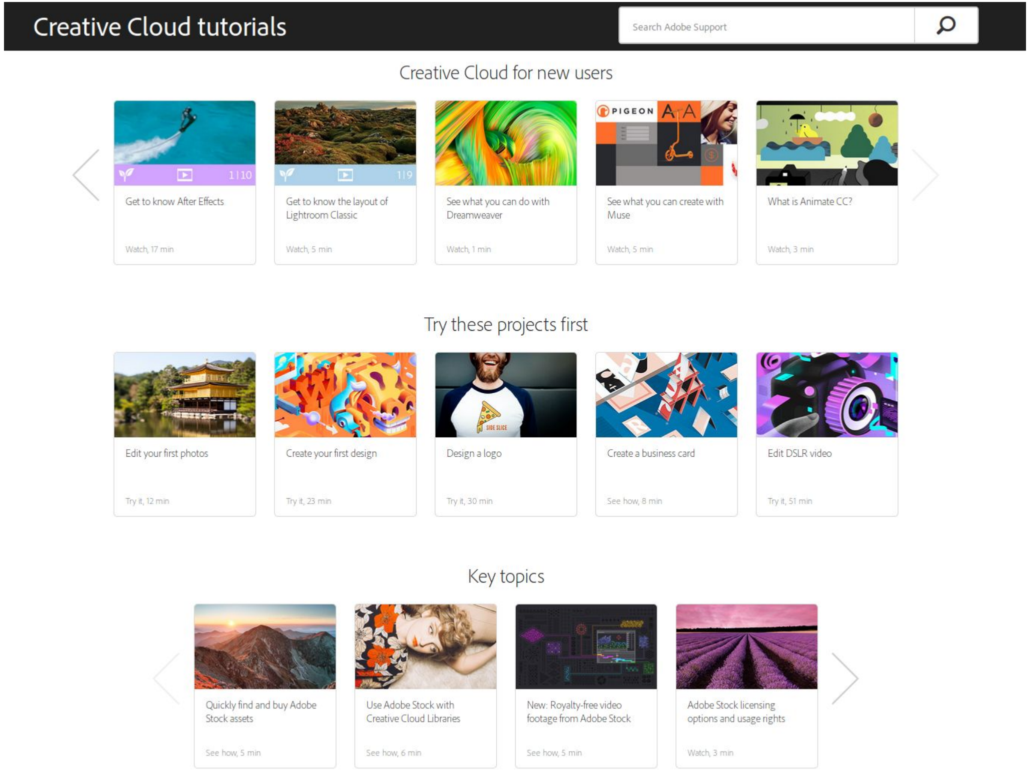
- Personalized tutoring systems
- Advertisement/marketing
- Medical treatment - drug prescription
- Portfolio management



Problem Statement

Thousands of possible actions!

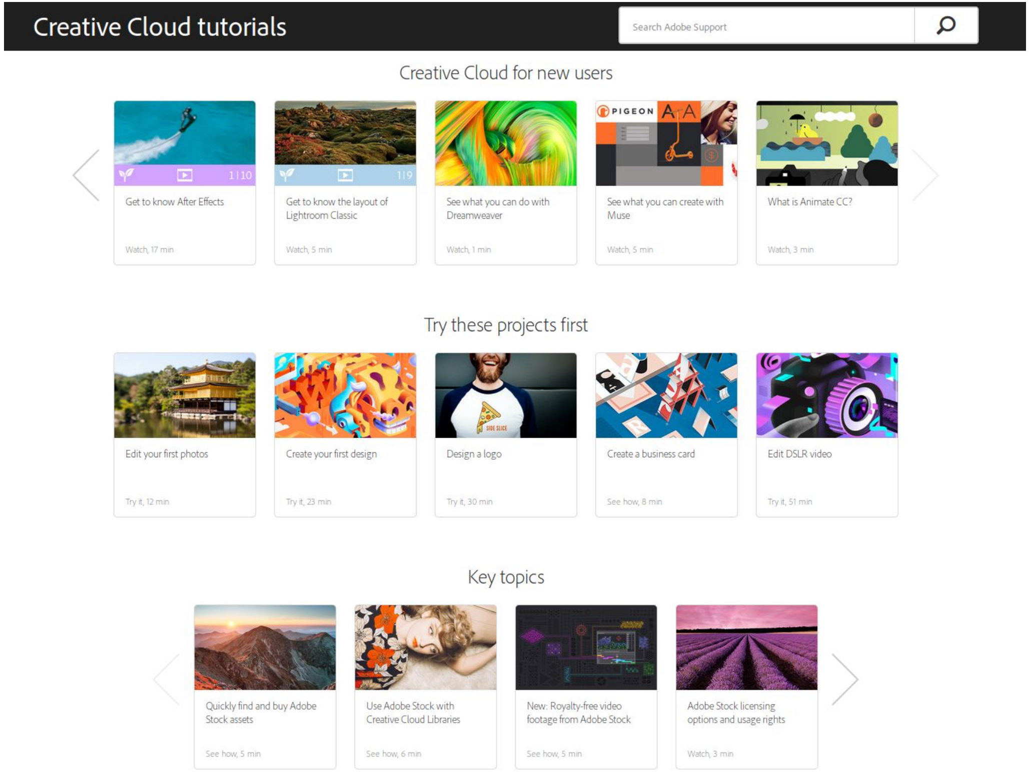
- Personalized tutoring systems
- Advertisement/marketing
- Medical treatment - drug prescription
- Portfolio management
- Video/Songs recommendation



Problem Statement

Thousands of possible actions!

- Personalized tutoring systems
- Advertisement/marketing
- Medical treatment - drug prescription
- Portfolio management
- Video/Songs recommendation
- …
- …
- Option selection

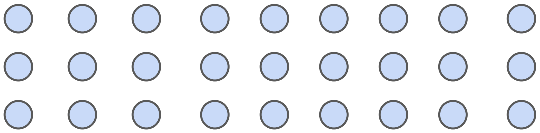


Problem Statement

Thousands of possible actions!

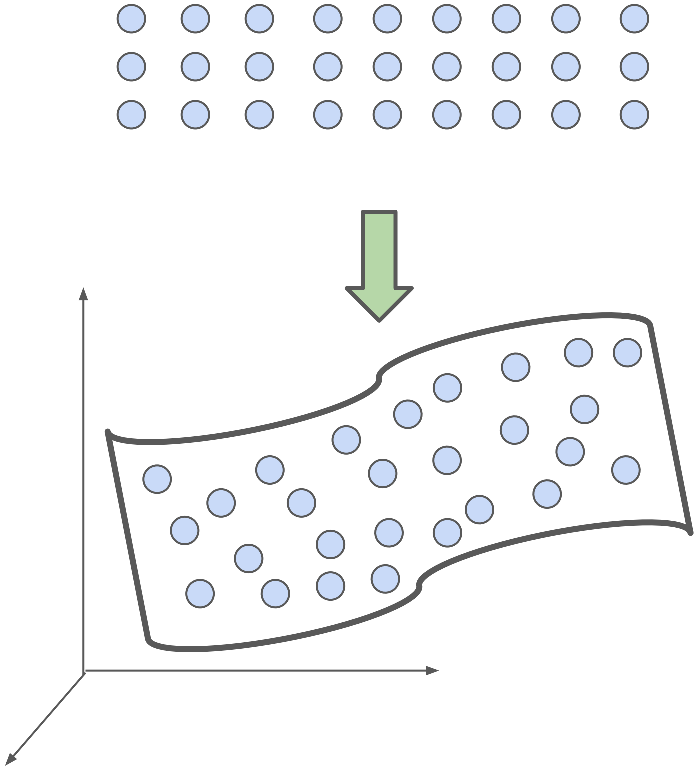
- Personalized tutoring systems
- Advertisement/marketing
- Medical treatment - drug prescription
- Portfolio management
- Video/Songs recommendation
- …
- …
- Option selection

Key Insights



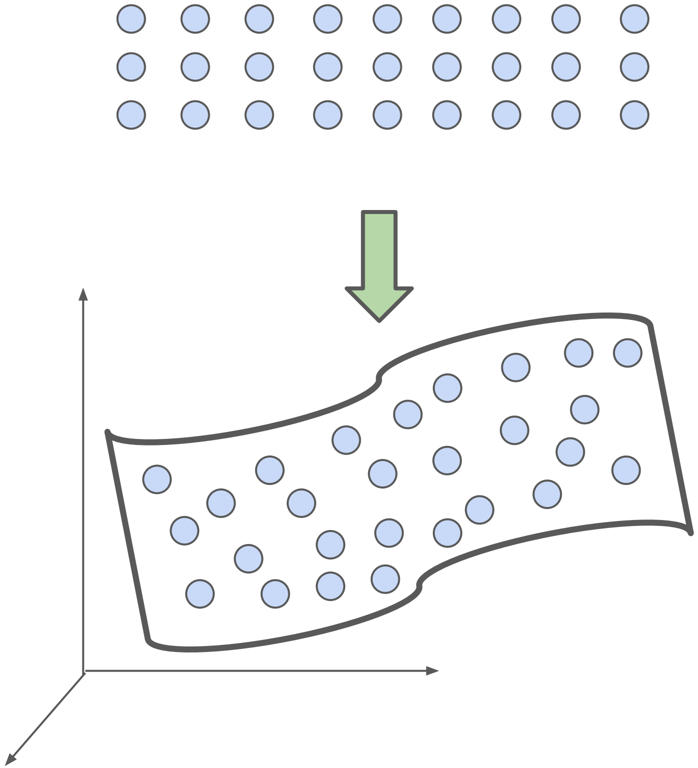
- Actions are **not independent** discrete quantities.

Key Insights



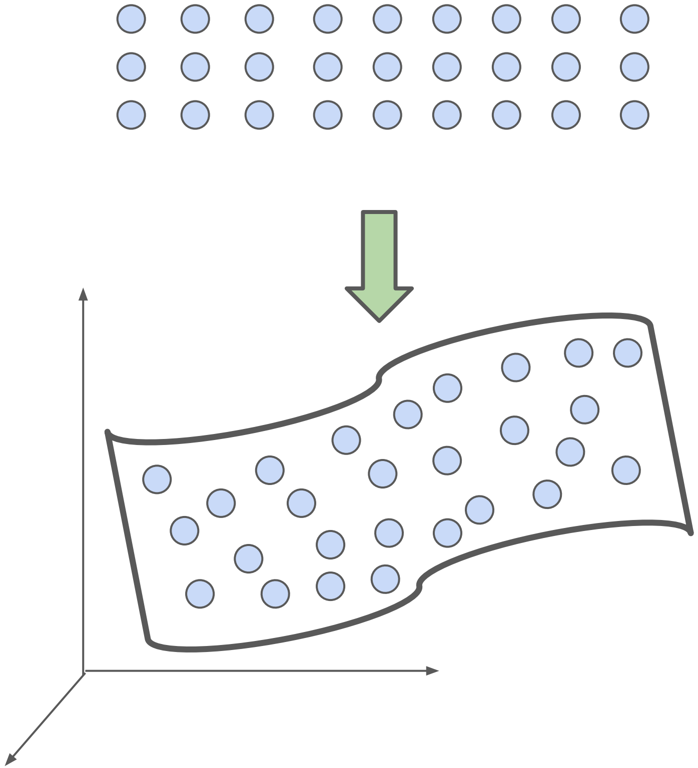
- Actions are **not independent** discrete quantities.
- There is a **low dimensional structure** underlying their behavior pattern.

Key Insights



- Actions are **not independent** discrete quantities.
- There is a **low dimensional structure** underlying their behavior pattern.
- This structure can be learned **independent of the reward**.

Key Insights



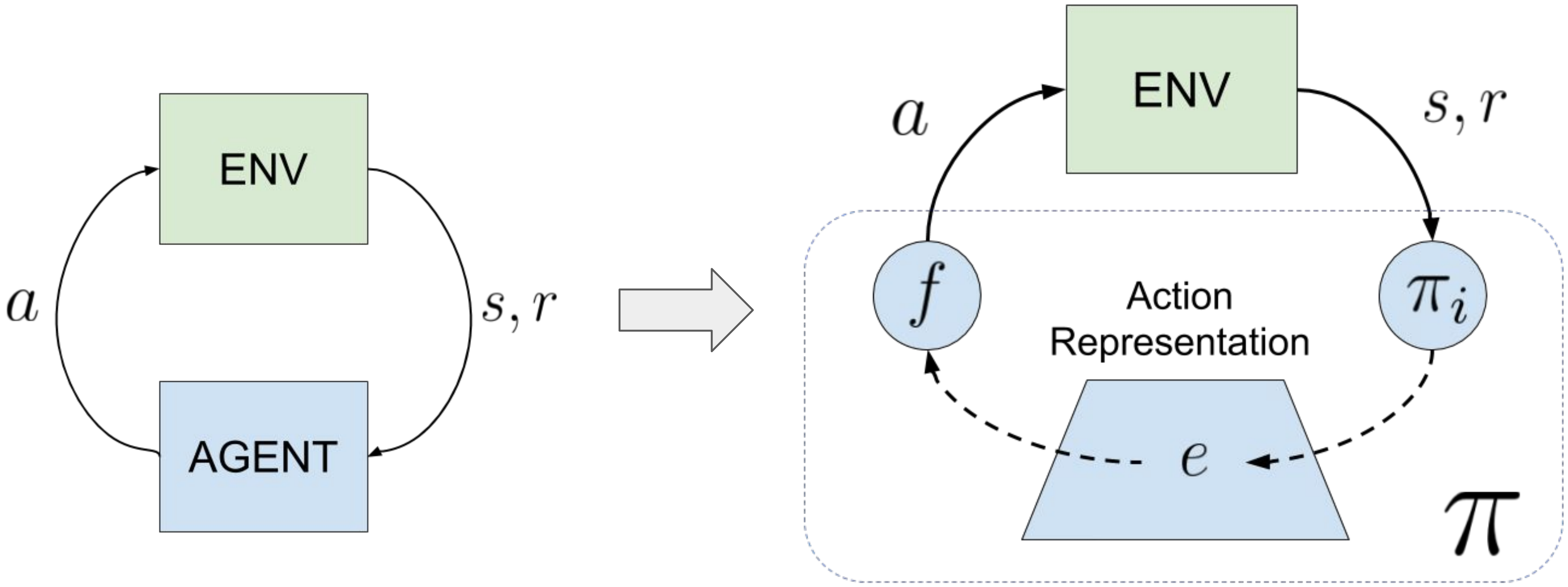
- Actions are **not independent** discrete quantities.

- There is a **low dimensional structure** underlying their behavior pattern.

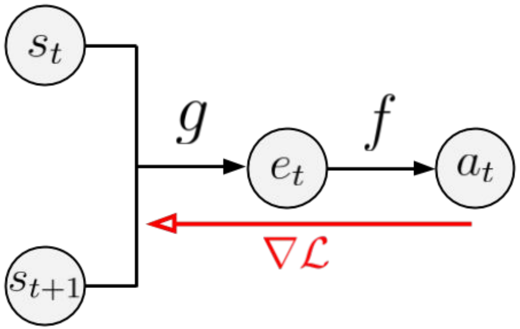
- This structure can be learned **independent of the reward**.

- Instead of raw actions, agent can **act in this space of behavior** and **feedback can be generalized** to similar actions.

Proposed Method



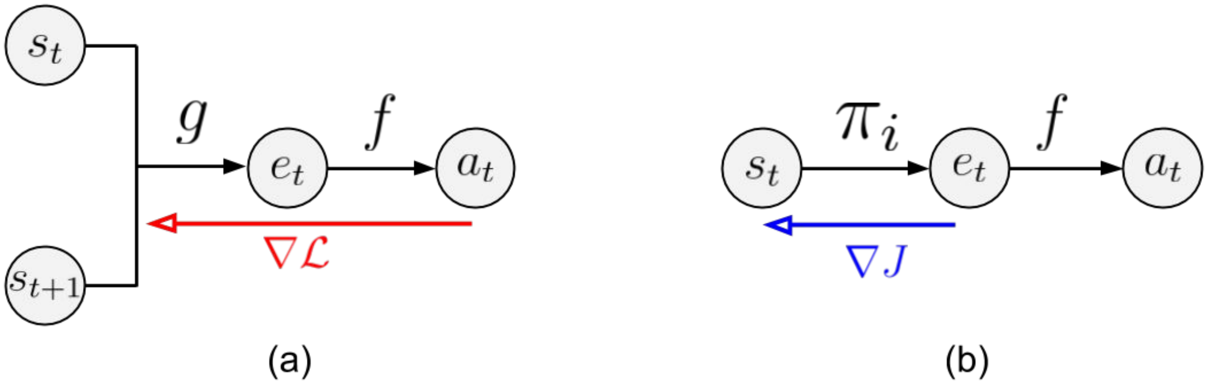
Algorithm



(a)

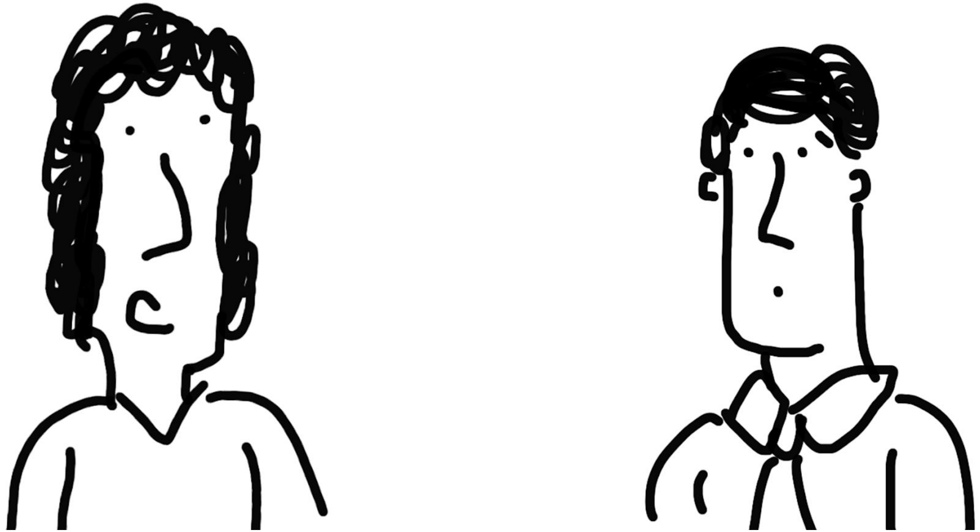
(a) Supervised learning of action representations.

Algorithm

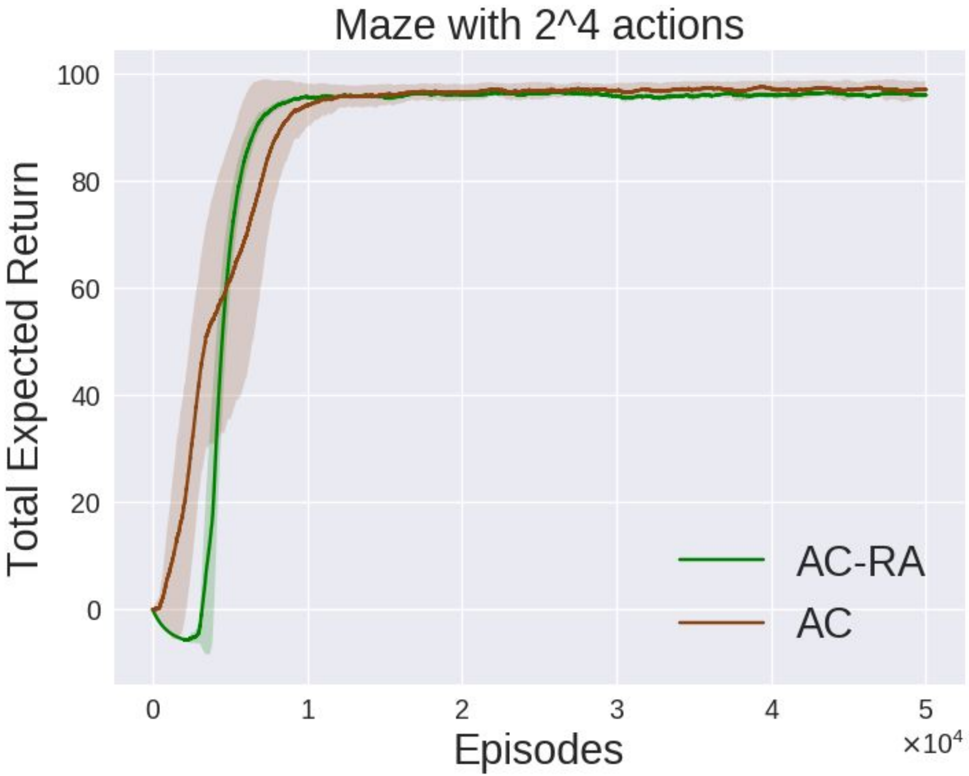


(a) Supervised learning of action representations.
(b) Learning internal policy with policy gradients.

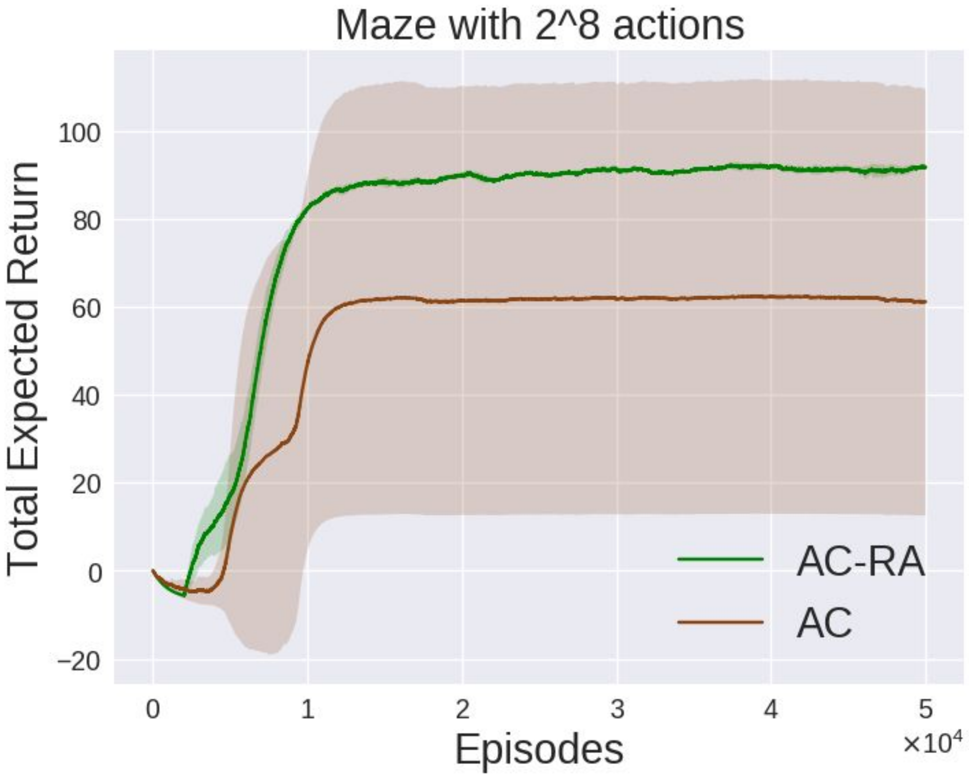
So, did it work?



Results

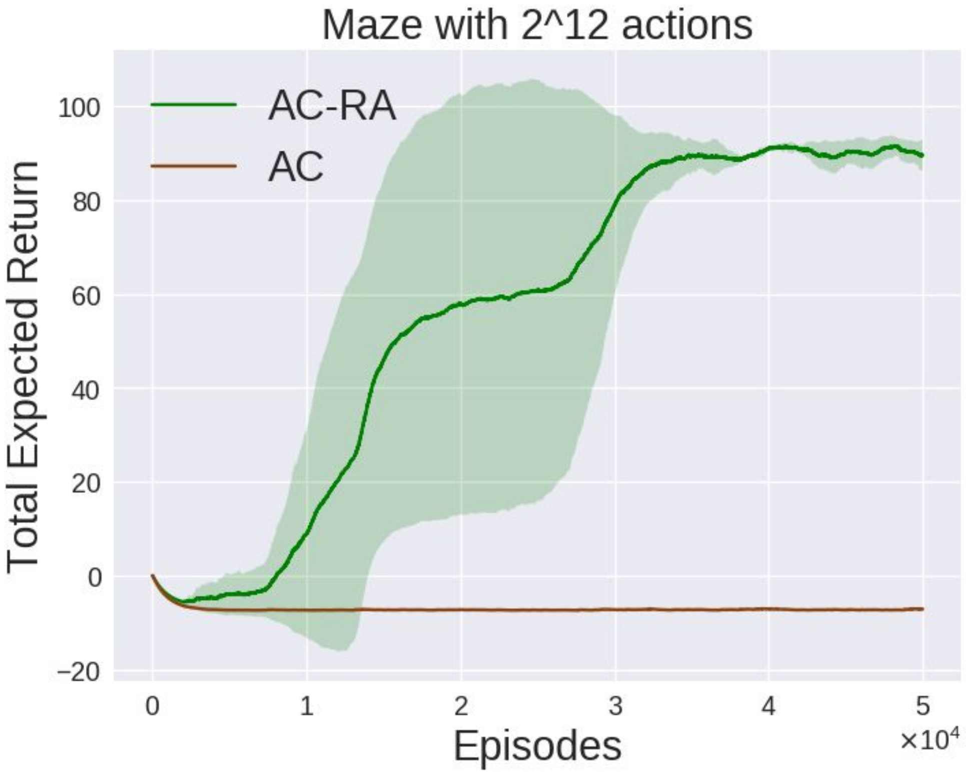

Maze with 2^4 actions
Total Expected Return
100
80
60
40
20
0
0
1
2
3
4
5
×10^4
Episodes
AC-RA
AC

Results



Maze with 2^8 actions
Total Expected Return
100
80
60
40
20
0
−20
0
1
2
3
4
5
×10^4
Episodes
AC-RA
AC

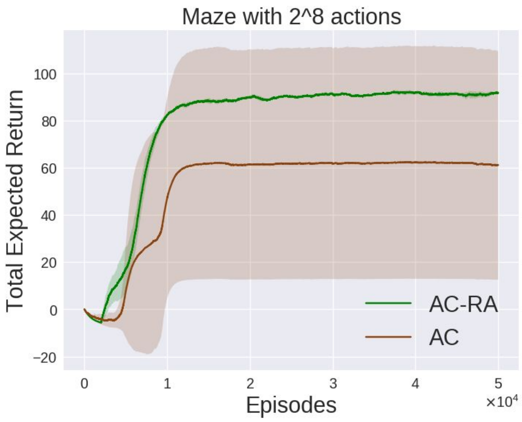
Results



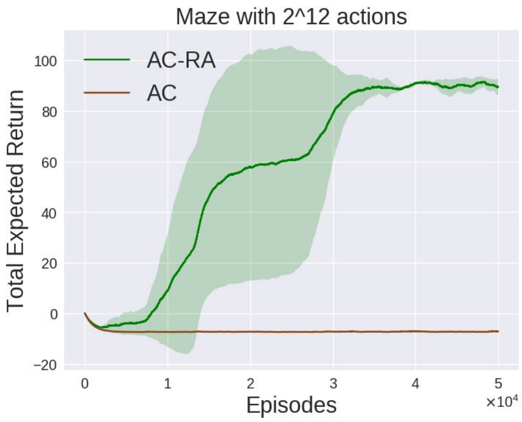
Maze with 2^12 actions
AC-RA
AC
Total Expected Return
100
80
60
40
20
0
−20
0
1
2
3
4
5
×10⁴
Episodes

Results

Maze with 2^4 actions
Total Expected Return
Episodes
×10^4
AC-RA
AC



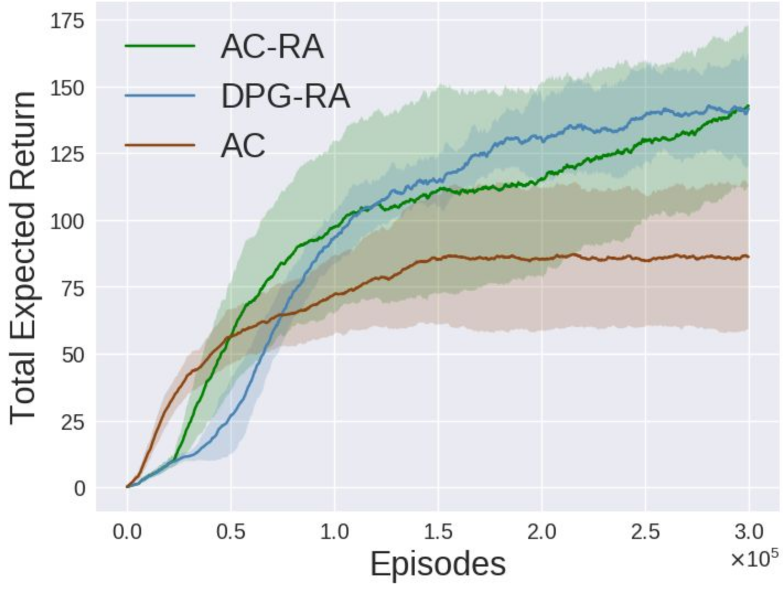
Maze with 2^8 actions
Total Expected Return
Episodes
×10^4
AC-RA
AC



Maze with 2^12 actions
Total Expected Return
Episodes
×10^4
AC-RA
AC

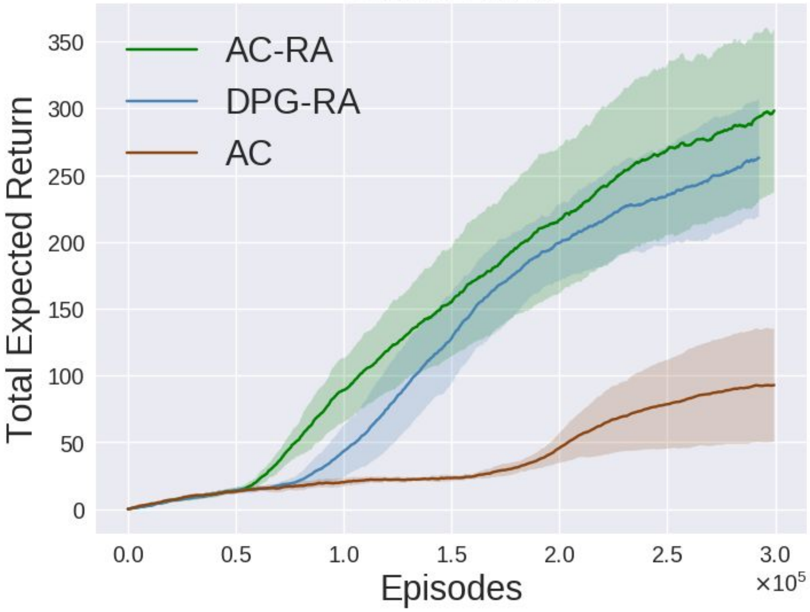
Real-world Applications at Adobe

HelpX



Actions = 1498 tutorials

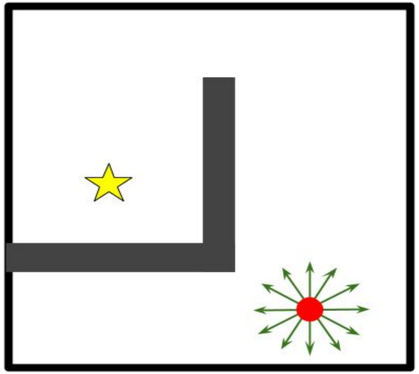
Photoshop



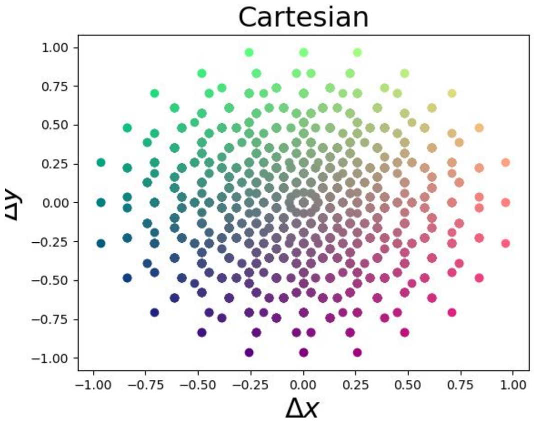
Actions = 1843 tools

Poster
#112
Today

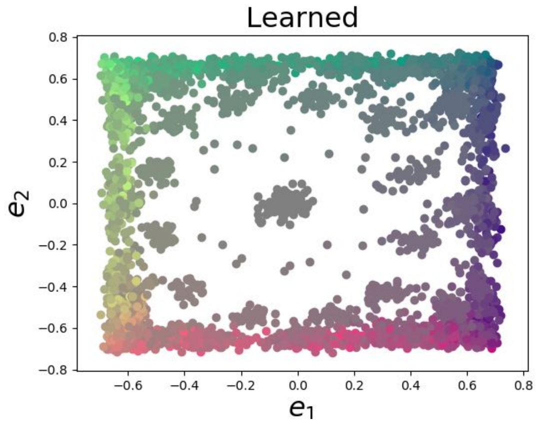
Results (Action representations)



Maze domain



Actual behavior of 2^{12} actions



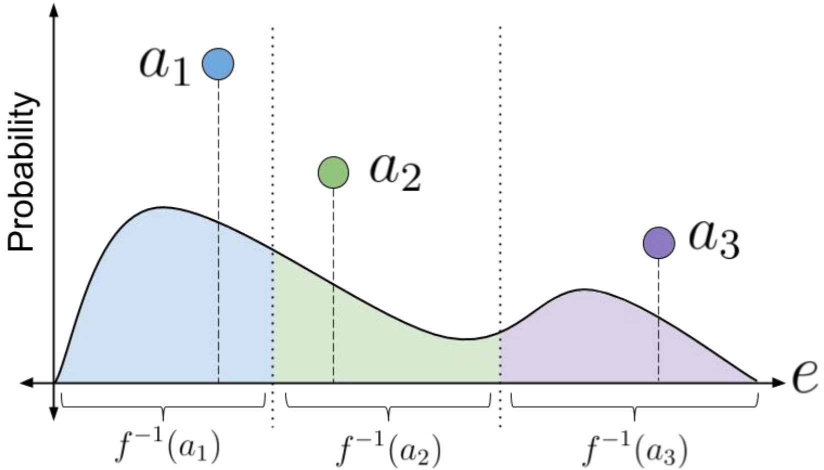
Learned representations of 2^{12} actions

Policy decomposition

We can now consider a new overall policy, π_o, such that

$$\pi_o(a|s) = \int_{f^{-1}(a)} \pi_i(e|s)\mathrm{d}e.$$

Here, $\pi_i(e|s)$ represents a new internal policy which selects the action representation for the given state s.



Case 1: Action representations are known

- The internal policy acts in the space of action representations

- Any existing policy gradient algorithm can be used to improve its local performance, independent of the mapping function.

Property 1. *For a deterministic function, f, that maps each point, $e \in \mathbb{R}^{d_e}$, in the representation space to an action, $a \in \{0,1\}^{|\mathcal{A}|}$, the expected updates to θ based on $\nabla J_i(\theta)$ are equivalent to updates based on $\nabla J_o(\theta, f)$. That is,*

$$\nabla J_o(\theta, f) = \nabla J_i(\theta).$$

Case 2: Learning action representations

- $P(a|e)$ required to map representation to action can be learned by satisfying the earlier assumption:

$$P(a|s,s') = \int_e P(a|e)P(e|s,s')$$

- We parameterize $P(a|e)$ and $P(e|s,s')$ with learnable functions ***f*** and ***g***, respectively.

- Observed transition tuples are from the required distribution.

- Parameters can be learned by minimizing the stochastic KL divergence.

- Procedure is **independent of reward**.

Experiments

Toy Maze:

- Agent in continuous state with n actuators.
- 2^n actions. Exponentially large action space.
- Long horizon and single goal reward.

Adobe Datasets:

- N-gram based multi-time step user behavior model from passive data.
- Rewards defined using a surrogate objective.
- **Photoshop** tool recommendation (**1843** tools)
- **HelpX** tutorial recommendation (**1498** tutorials)

Advantages

- **Exploits structure** in space of actions.
- **Quick generalization of feedback** to similar actions.
- **Less parameters** updated using high variance policy gradients.
- **Drop-in extension** for existing policy gradient algorithms.