Reinforcement learning

Episode -1

Miscellaneous cool stuff

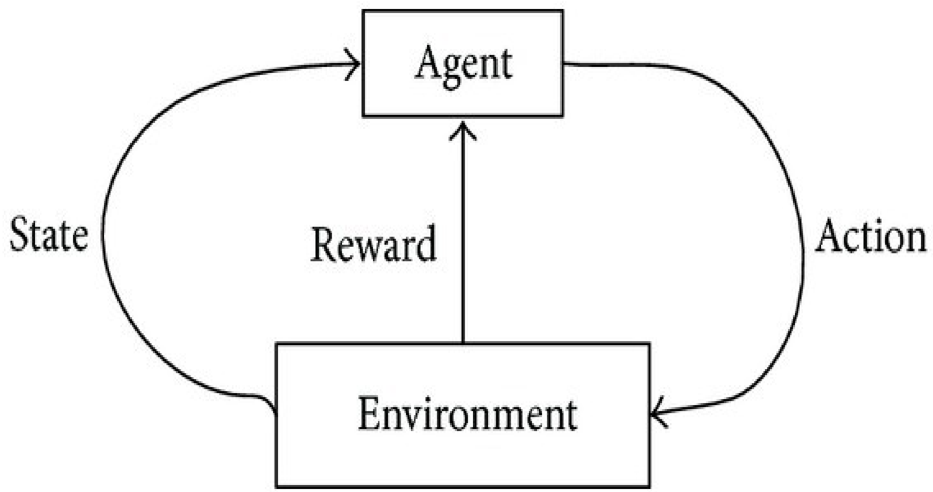
Yandex
Data Factory

LAMBDA

British Hedgehog
Preservation Society

Part 1: Continuous action spaces

- Regular MDP
- $a \in \mathbb{R}^n$



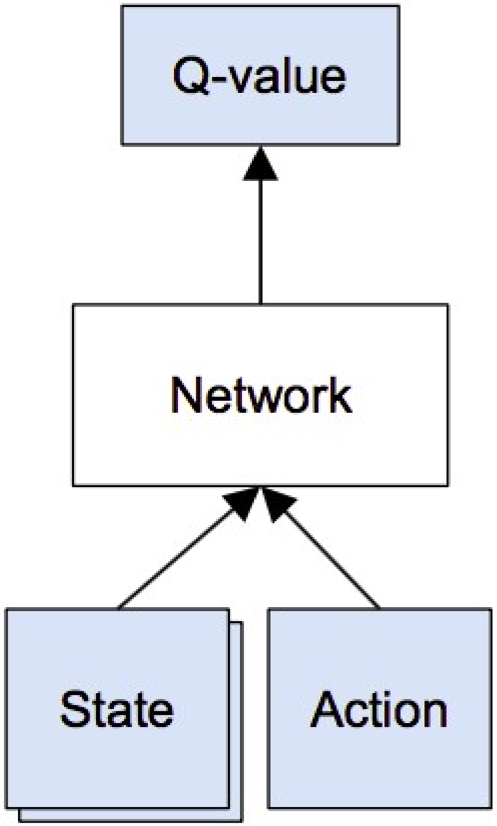
Which methods can we use?

Continuous action spaces

- We can learn critic easily

- The problem is finding

$$a_{opt}(s) = \underset{a}{argmax}\, Q(s,a)$$



Worst case: optimize over neural net!

Normalized advantage functions

Idea 1: restrict Q(s,a) so that optimization becomes trivial

For example, parabola (for 1d action space)

$$Q(s,a)=V(s)+A(s,a)$$

$$A(s,a)=-k_\theta(s)\cdot(a-\mu_\theta(s))^2$$

How to find optimal a?

Normalized advantage functions

Idea 1: restrict Q(s,a) so that optimization becomes trivial

For example, parabola (for 1d action space)

$$Q(s,a)=V(s)+A(s,a)$$

$$A(s,a)=-k_\theta(s)\cdot(a-\mu_\theta(s))^2$$

How to find optimal a? - a_opt = mu(s)

Normalized advantage functions

Idea 1: restrict Q(s,a) so that optimization becomes trivial

For example, parabola (for 1d action space)

$$Q(s,a) = V(s) + A(s,a)$$

$$A(s,a) = -k_\theta(s) \cdot (a - \mu_\theta(s))^2$$

How does it generalize for n-dimensional **a**?

Normalized advantage functions

Idea 1: restrict Q(s,a) so that optimization becomes trivial

For example, parabola (for 1d action space)

$$Q(s,a)=V(s)+A(s,a)$$

$$A(s,a)=-0.5\cdot(a-\mu_\theta(s))^T\cdot L(s)\cdot L(s)^T(a-\mu_\theta(s))$$

Where L(s) is a lower-triangular matrix

Normalized advantage functions

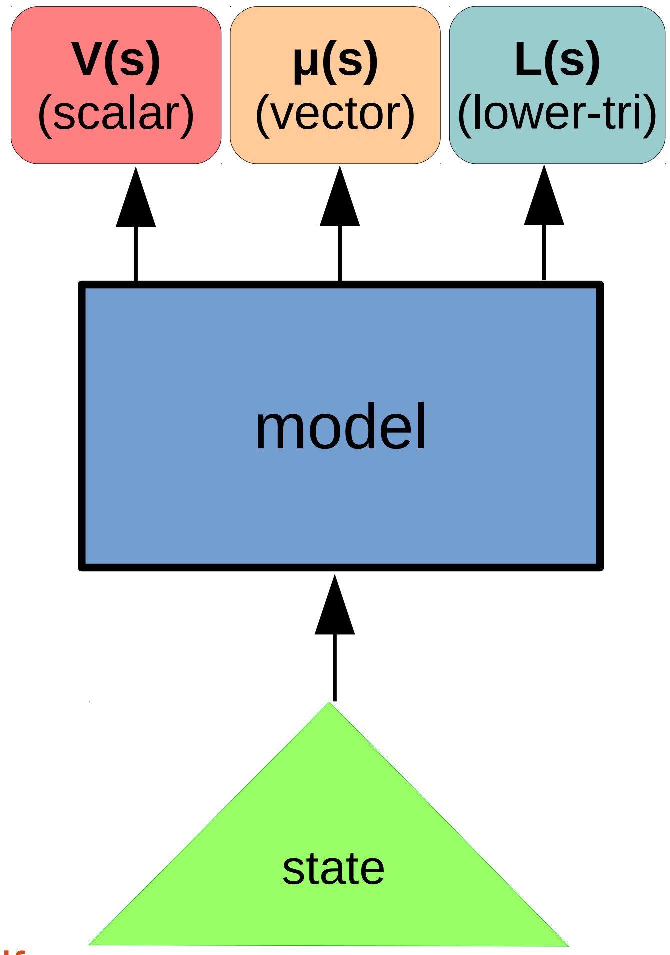
Network:

(trains end-to-end)

$$Q(s,a) = V(s) + A(s,a)$$

$$A(s,a) = \dots$$

$$\underset{\theta}{argmin}\left(Q(s_t,a_t) - [r + \gamma \cdot V(s_{t+1})]\right)^2$$



https://arxiv.org/pdf/1603.00748.pdf

Deterministic policy gradient

- Idea2: learn a separate network to find a_opt

- Train critic $Q_\theta(s, a)$

$$\underset{\theta}{argmin}\left(Q(s_t, a_t) - [r + \gamma \cdot V(s_{t+1})]\right)^2$$

- Train actor $a_{opt}(s) \approx \mu_\theta(s)$

$$\nabla_\theta J = \frac{\partial Q^\theta(s,a)}{\partial a} \frac{\partial \mu(s|\theta)}{\partial \theta}$$

Deterministic policy gradient

- Idea2: learn a separate network to find a_opt

- Train critic $Q_\theta(s, a)$

$$\underset{\theta}{argmin}\left(Q(s_t, a_t) - [r + \gamma \cdot V(s_{t+1})]\right)^2$$

How do we get V(s')?

- Train actor $a_{opt}(s) \approx \mu_\theta(s)$

$$\nabla_\theta J = \frac{\partial Q^\theta(s,a)}{\partial a} \frac{\partial \mu(s|\theta)}{\partial \theta}$$

Deterministic policy gradient

- Idea2: learn a separate network to find a_opt

- Train critic $Q_\theta(s, a)$

$$\underset{\theta}{argmin}\left(Q(s_t, a_t) - [r + \gamma \cdot Q(s_{t+1}, \mu_\theta(s_{t+1}))]\right)^2$$

- Train actor $a_{opt}(s) \approx \mu_\theta(s)$

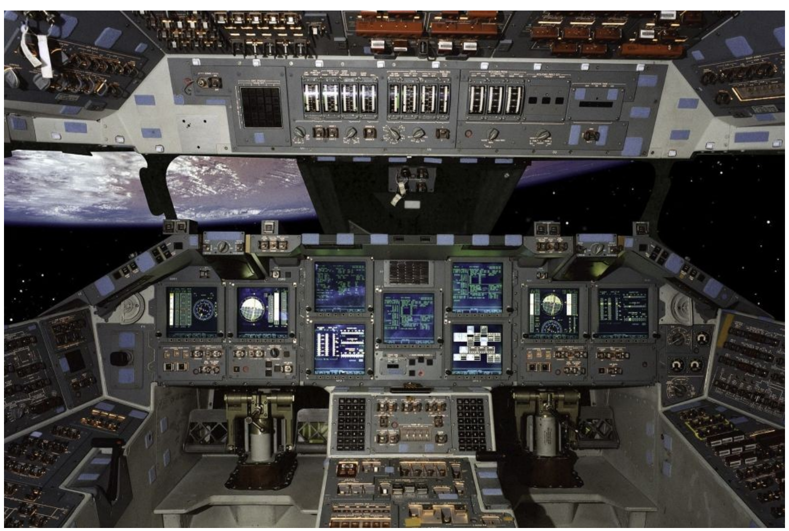
$$\nabla_\theta J = \frac{\partial Q^\theta(s,a)}{\partial a} \frac{\partial \mu(s|\theta)}{\partial \theta}$$

Deterministic policy gradient

Demo with torcs http://bit.ly/2pXwdKa

Ugly action spaces

- Keyboard
- Aircraft control
- ~Any strategy game

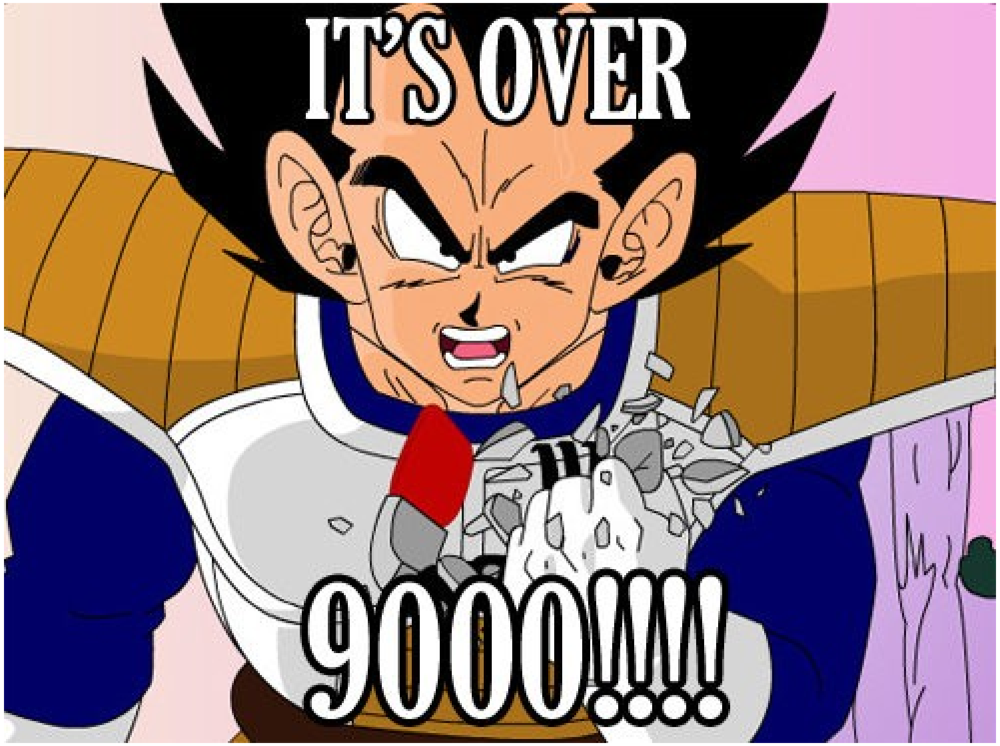


Exploiting action space structure

- Imagine a large discrete action space, 10^90

Exploiting action space structure

- Imagine a large discrete action space, 10^90



Like, way over 9000

Exploiting action space structure

- Imagine a large discrete action space, 10^90
  - We worked with such spaces before!

$$a = \langle a_0, a_1, a_2, \ldots, a_{50} \rangle$$

$$|a_i| = 60 \qquad\qquad |a| = 60^{50} \approx 10^{90}$$

Does it remind you of anything?

Exploiting action space structure

- Imagine a large discrete action space, 10^90
  - We worked with such spaces before!

$$a = \langle a_0, a_1, a_2, \dots, a_{50} \rangle$$

$$|a_i| = 60 \qquad\qquad |a| = 60^{50} \approx 10^{90}$$

a_i is a single letter, a is full translation

How did we deal with 10^90 actions?

Structured action space: sequential

- Action space

$$a=\langle a_0, a_1, a_2, \ldots, a_{60}\rangle$$

- Straightforward $\pi(a|s)$ $|s|\times|a|$

- Sequential

$$\pi(a|s)=\pi(a_0|s)\cdot\pi(a_1|s,a_0)\cdot\ldots\cdot\pi(a_i|s,a_{0:i-1})$$

Structured action space: sequential

- Action space

$$a = \langle a_0, a_1, a_2, \ldots, a_{60} \rangle$$

- Straightforward $\pi(a|s)$ $|s| \times |a|$

- Sequential

$$\pi(a|s) = \pi(a_0|s) \cdot \pi(a_1|s, a_0) \cdot \ldots \cdot \pi(a_i|s, a_{0:i-1})$$

How did we approximate this distribution?

Structured action space: sequential

- Action space

$$a = \langle a_0, a_1, a_2, \ldots, a_{60} \rangle$$

- Straightforward $\pi(a|s)$ $|s| \times |a|$

- Sequential

$$\pi(a|s) = \pi(a_0|s) \cdot \pi(a_1|s, a_0) \cdot \ldots \cdot \pi(a_i|s, a_{0:i-1})$$

How did we approximate this distribution?
With RNNs/HMMs!

Recommender systems

- Typical bandit
- One action per item, 10^5 items (still over 9000)
- Even worse: new items arrive over time!
- Similar items have similar audiences

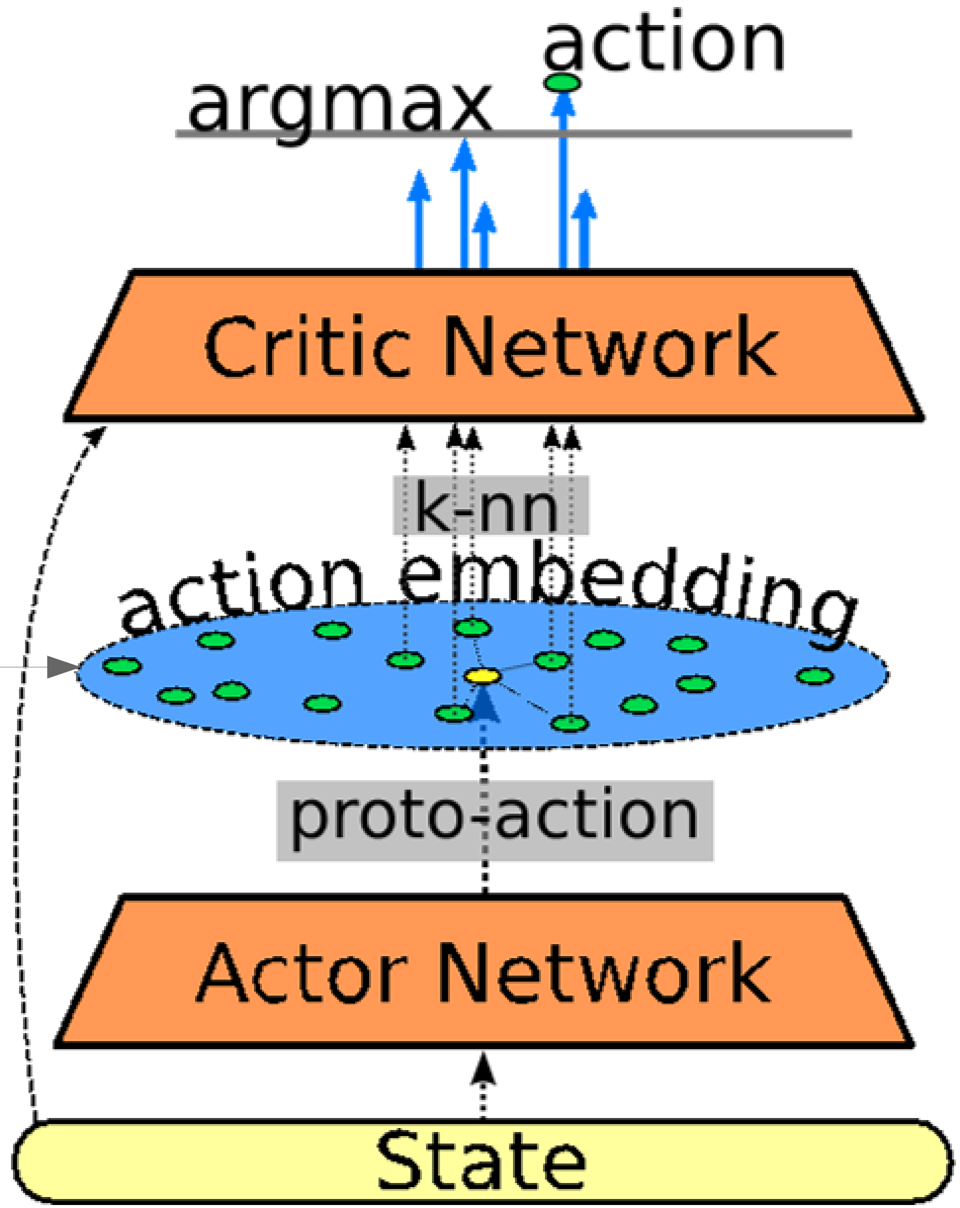
Recommender systems

- Typical bandit
- One action per item, 10^5 items (still over 9000)
- Even worse: new items arrive over time!
- Similar items have similar audiences

Action embedding

Map actions into some continuous space

How to do that for films?



Action embedding

- Map actions into some continuous space

  (hand-crafted / tsne / emb)

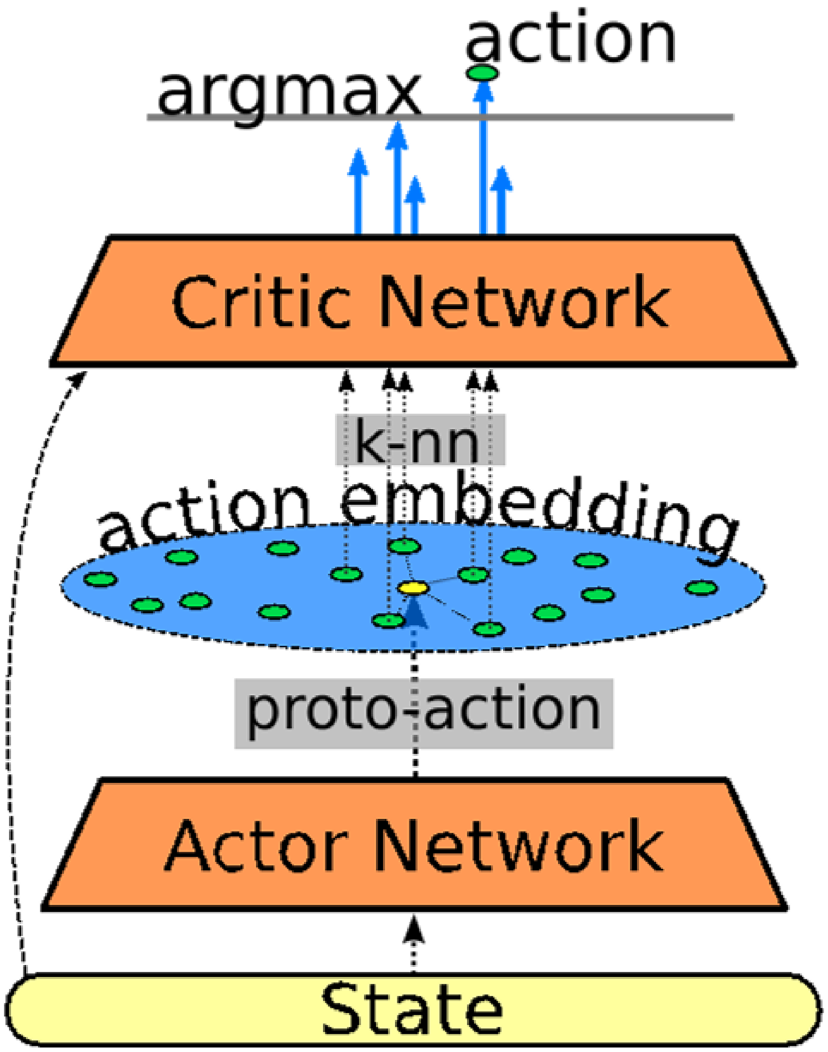
- Predict continuous “action”

  (aka proto-action)

- Pick closest discrete action

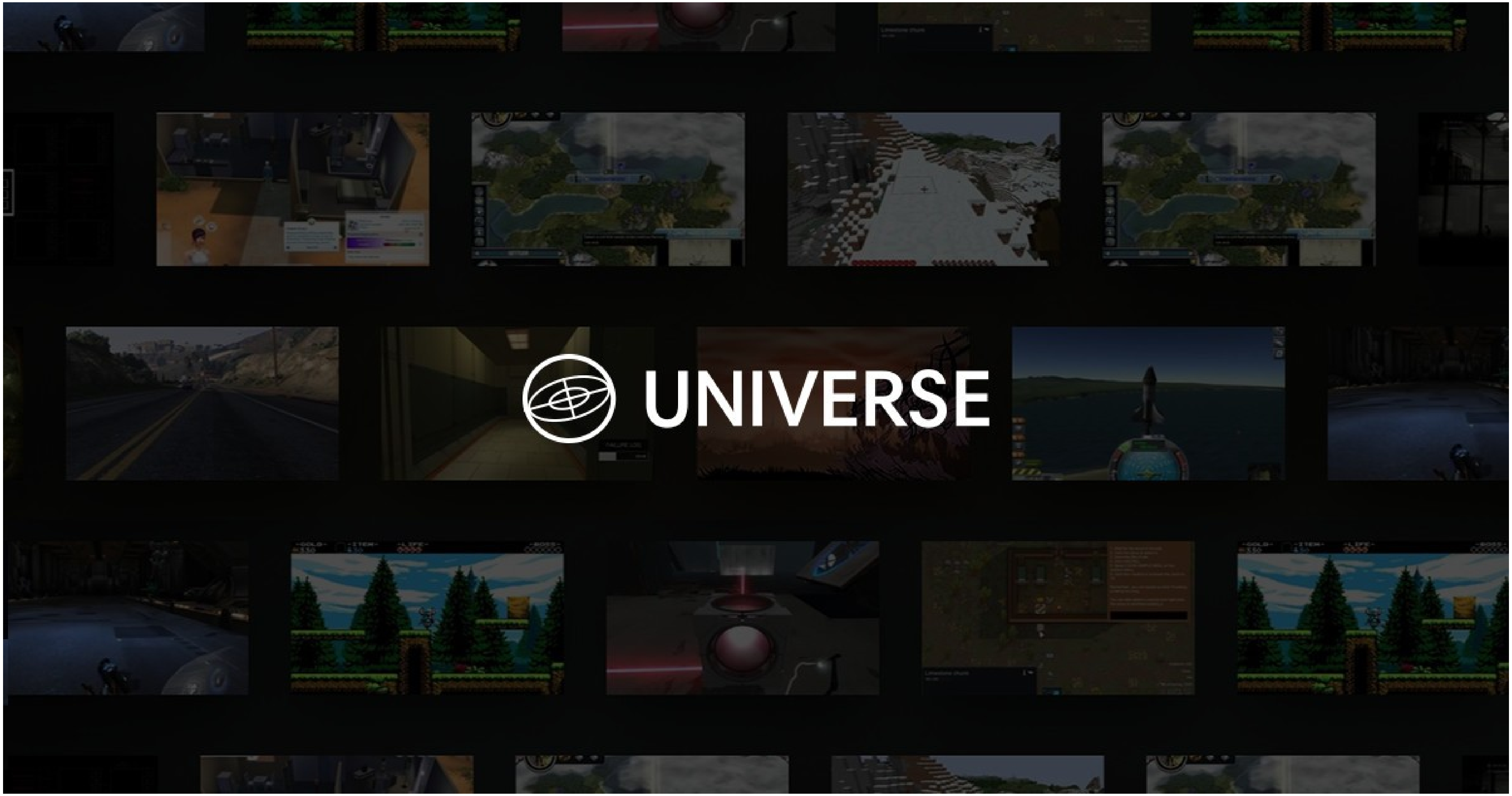
  e.g. with (LSH)

  (locally sensitive hashes)



https://arxiv.org/abs/1512.07679

Let's figure it out!



http://universe.openai.com

Let's figure it out!

- Emulate human PC user
  - Actions: keyboard and mouse

How do we define $\pi(a|s)$?

OpenAI

Part 2: hierarchical RL

Problem: Rewards are usually sparse (temporally rare) and delayed.

It takes exponentially more random exploration to learn optimal policy in case of rare rewards.

Humans:

- Don't seem to follow epsilon-greedy exploration policy (see lecture 9)

- Think in several layers of abstraction
  - “Contract leg muscles”
  - “Push gas pedal (while driving)”
  - “Take left turn in 15 meters”
  - “Drive to school”,
  - “Give my children education”

Humans:

- Don't seem to follow epsilon-greedy exploration policy (see lecture 9)

- Think in several layers of abstraction
  - “Contract leg muscles”
  - “Push gas pedal (while driving)”
  - “Take left turn in 15 meters”
  - “Drive to school”,
  - “Give my children education”

We want hierarchical decision-making!

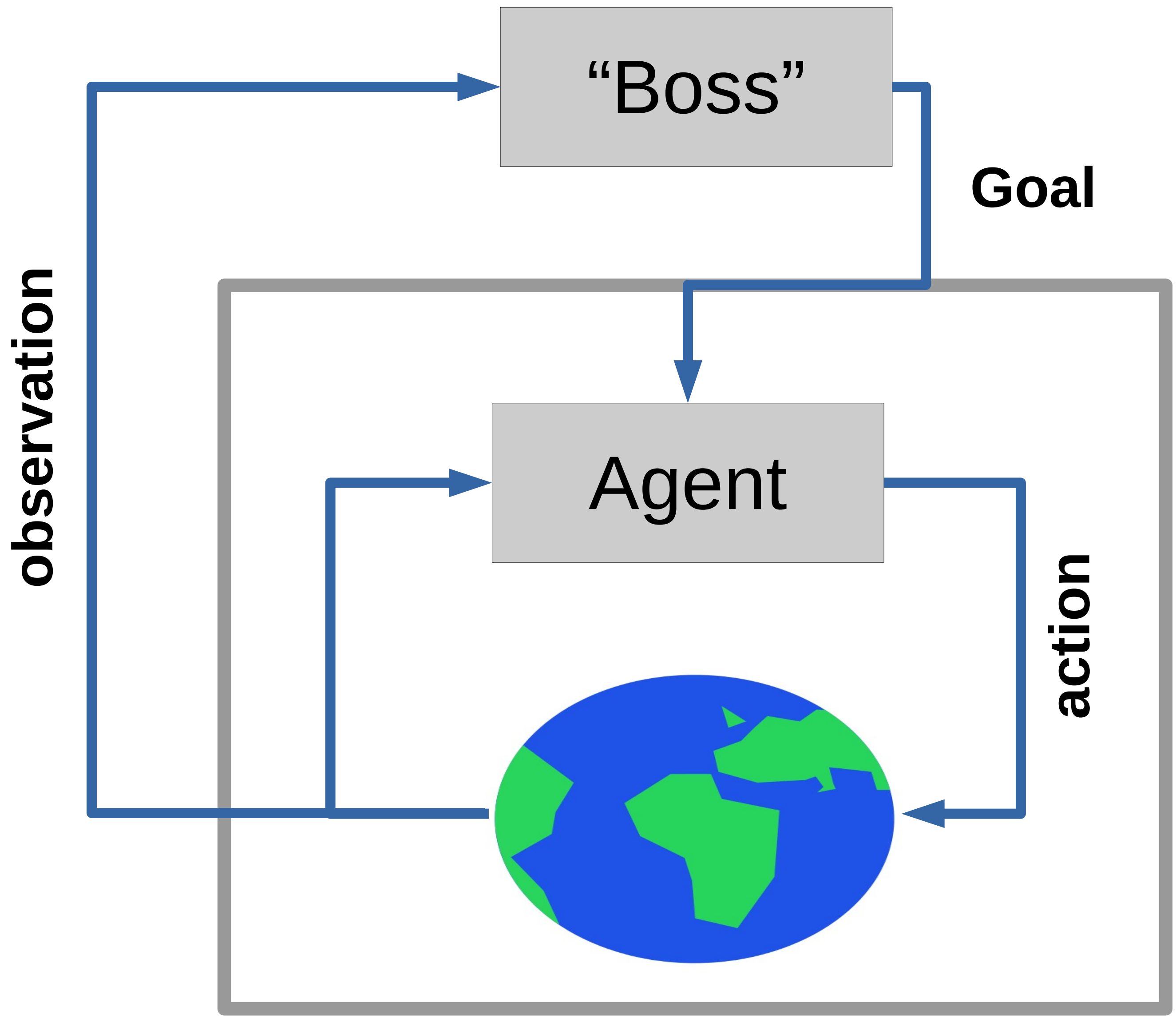
So what is hierarchy, again?

Suggestions?

So what is hierarchy, again?

- I know, I know! It's about operating in term of more abstract states!
- No! It's about acting on a longer time scale!
- No! it's about decomposing reward into short-term and long-term

Hierarchical MDP



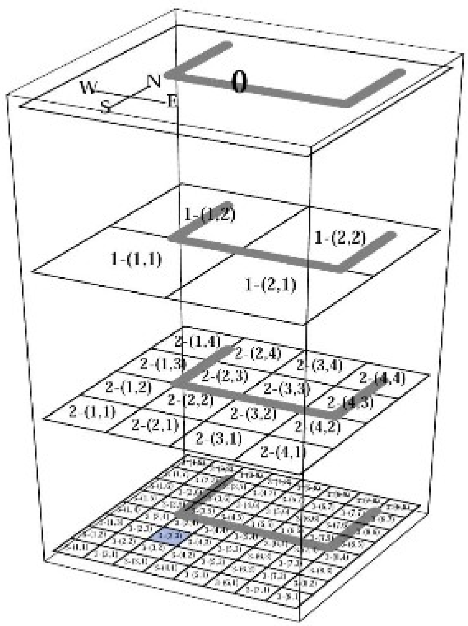
Feudal RL

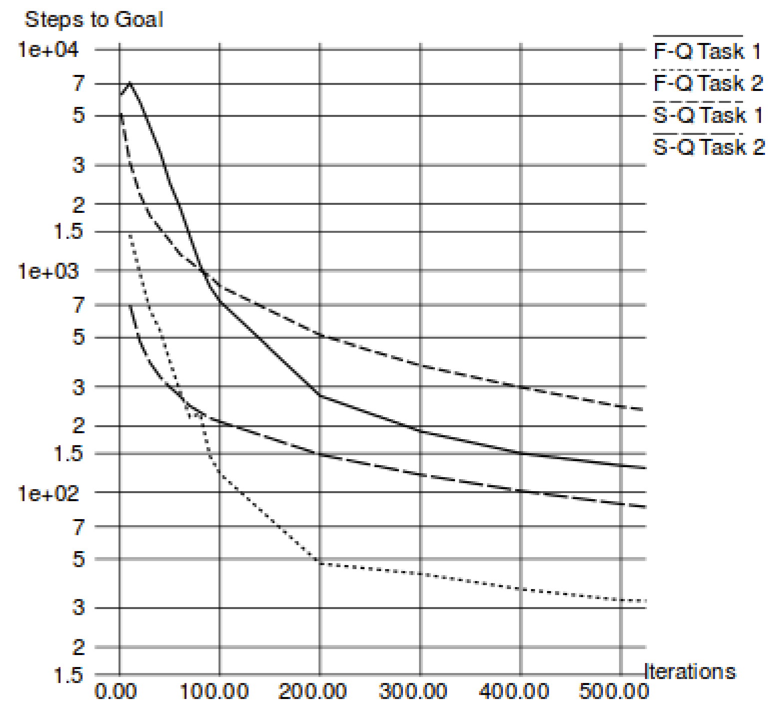
- Idea:
  - At the bottom, there's regular agent that operates on raw states and takes actual actions. He also receives goals from above.
  - Above him, there's a 'manager' agent that operates on more abstract states and issues goals to bottom agent. He also may receive higher-order goals from above him.

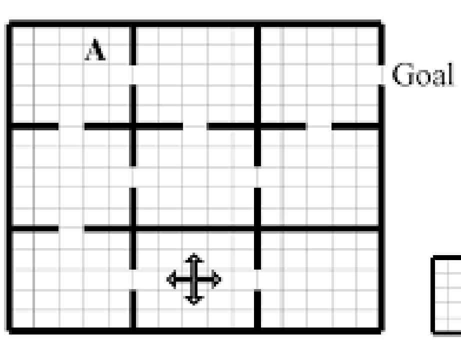
- Manager doesn't care whether his goal is actually beneficial to his super.
- Neither he does care about what orders does his sub-manager issue.

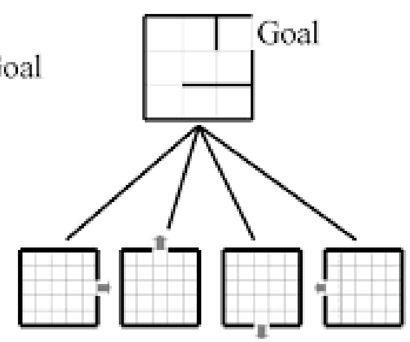
Feudal RL

- Idea:

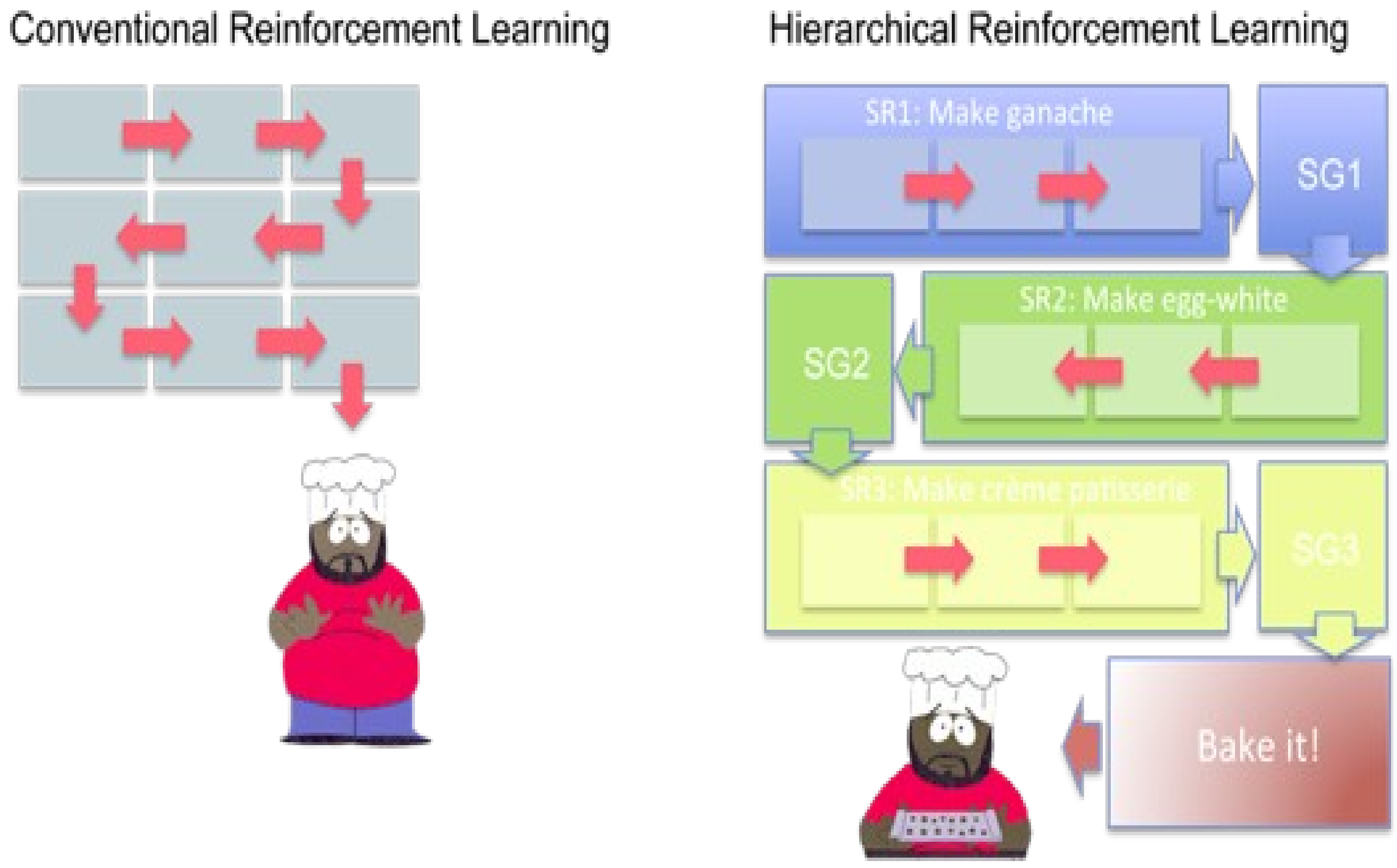






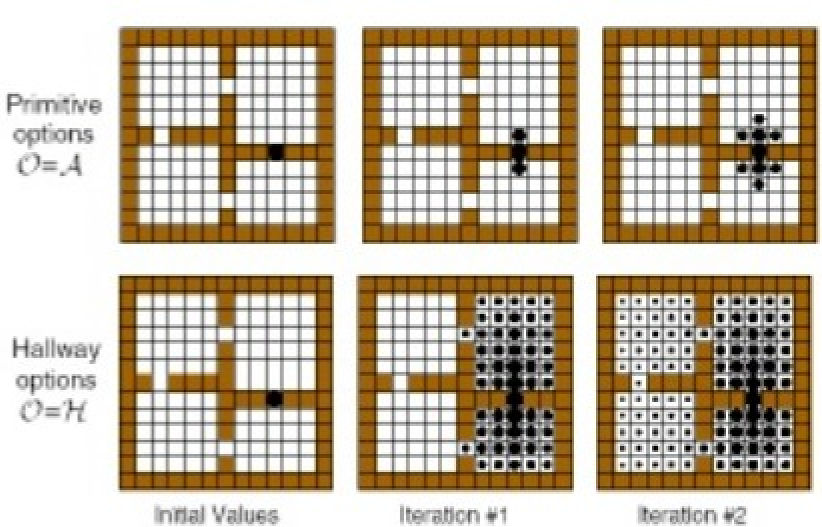


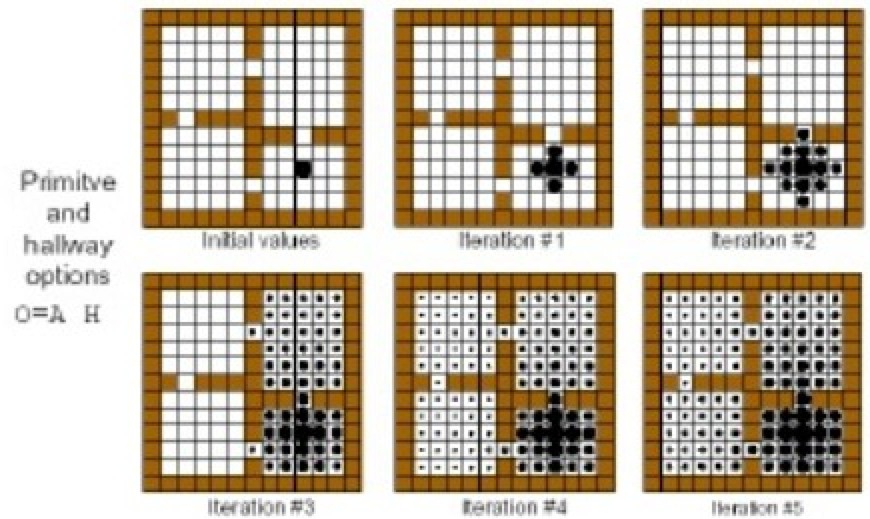
Temporal abstraction RL



The “option" method by Sutton, Precup, and Singh 1998

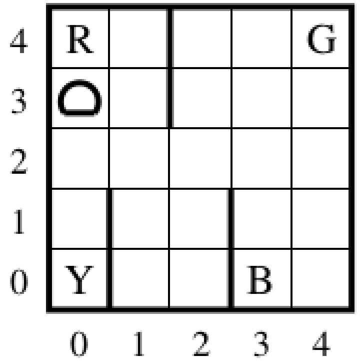
Temporal abstraction RL

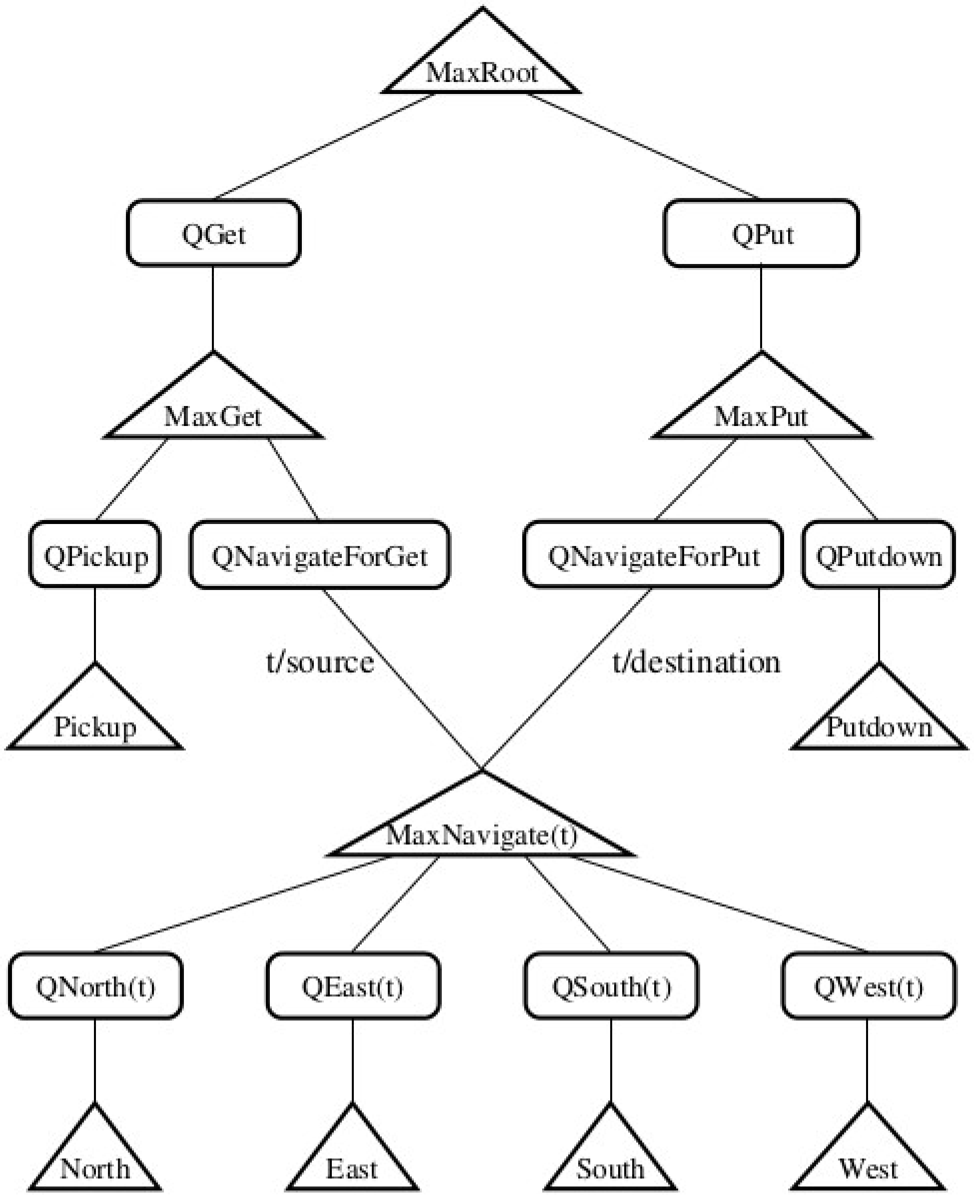




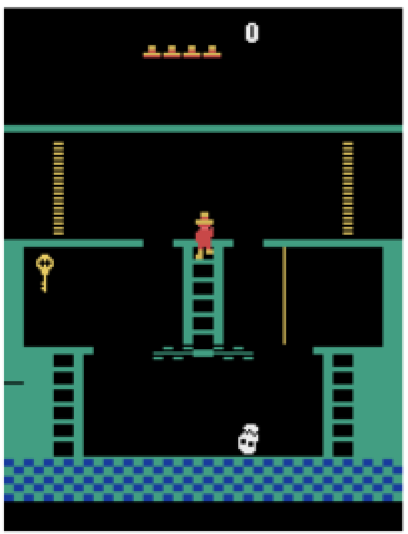
The “option" method by Sutton, Precup, and Singh 1998

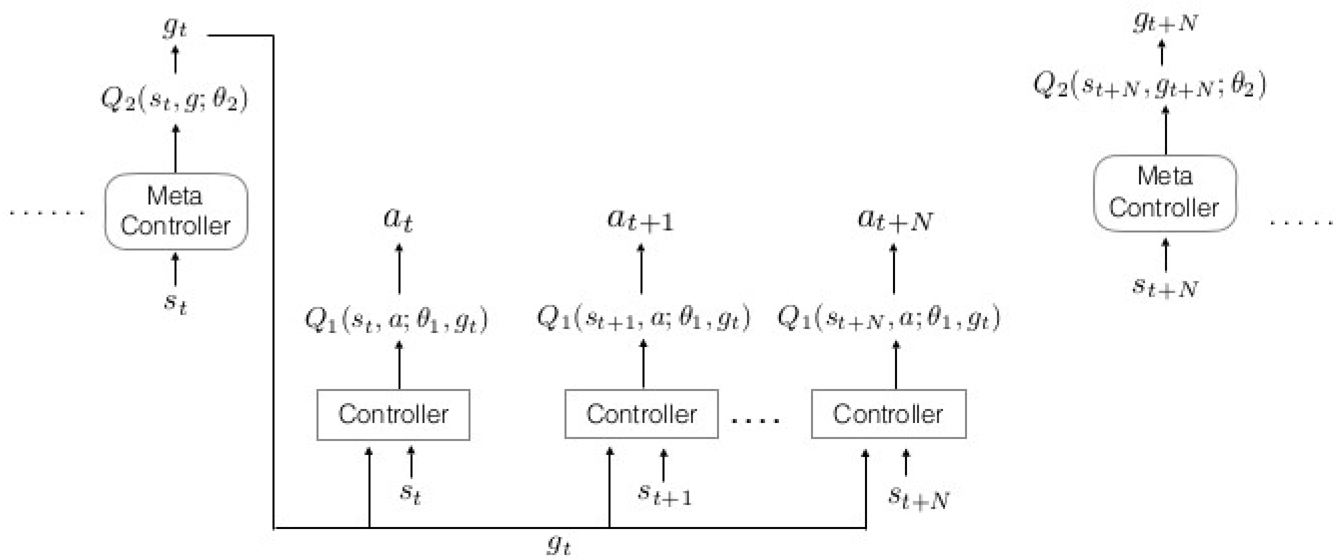
MDP decomposition





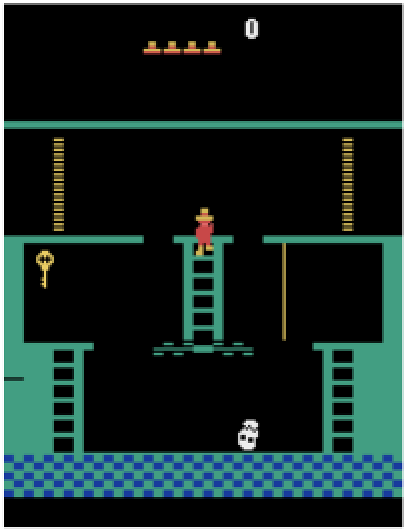
Hierarchical deep RL

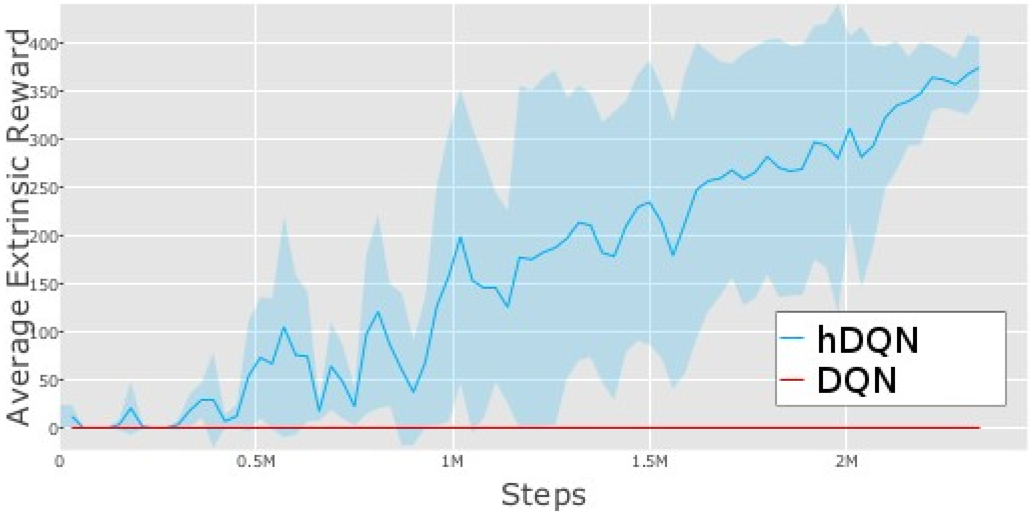




Tenenbaum et al. https://arxiv.org/abs/1604.06057v1

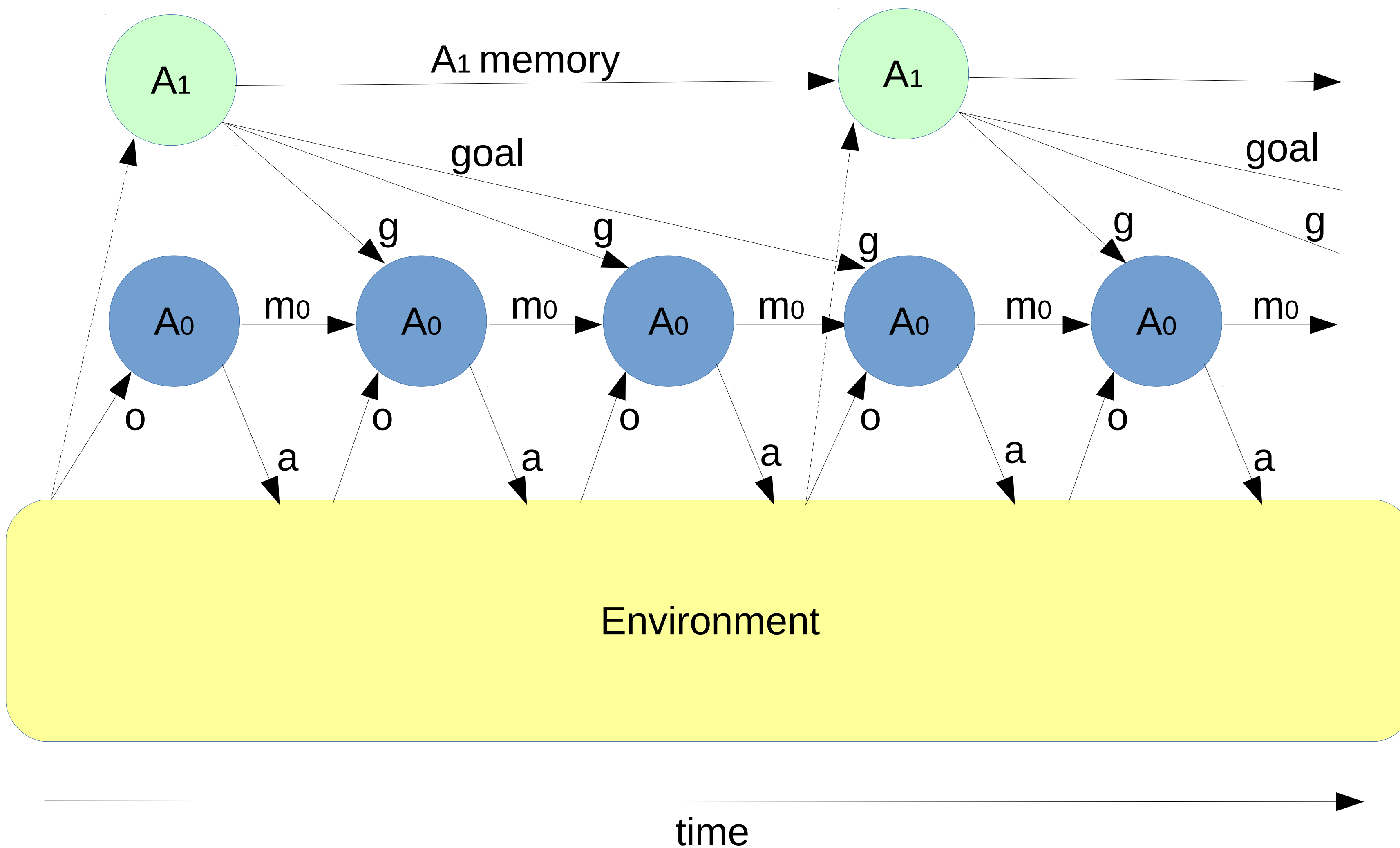
Hierarchical deep RL





Tenenbaum et al. https://arxiv.org/abs/1604.06057v1

Hierarchical RL



Model-free hierarchical RL

- Upper-level agent actions (goals for lower-level) are just arbitrary numbers without any model knowledge
- Lower-level agent uses goal embedding as a part of his state vector

$$A_1 : g \in N \pi\left(g \middle| s, m_1\right) : argmax\, R$$

Model-free hierarchical RL

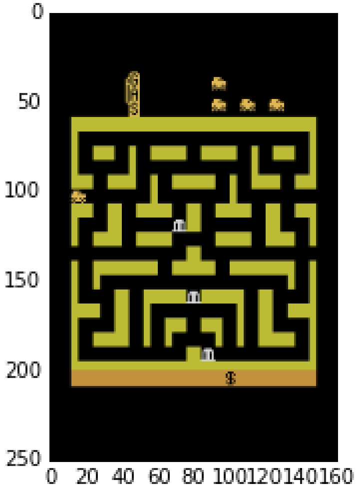
- Upper-level agent actions (goals for lower-level) are just arbitrary numbers without any model knowledge $g \in N$
- Lower-level agent uses goal embedding as a part of his state vector

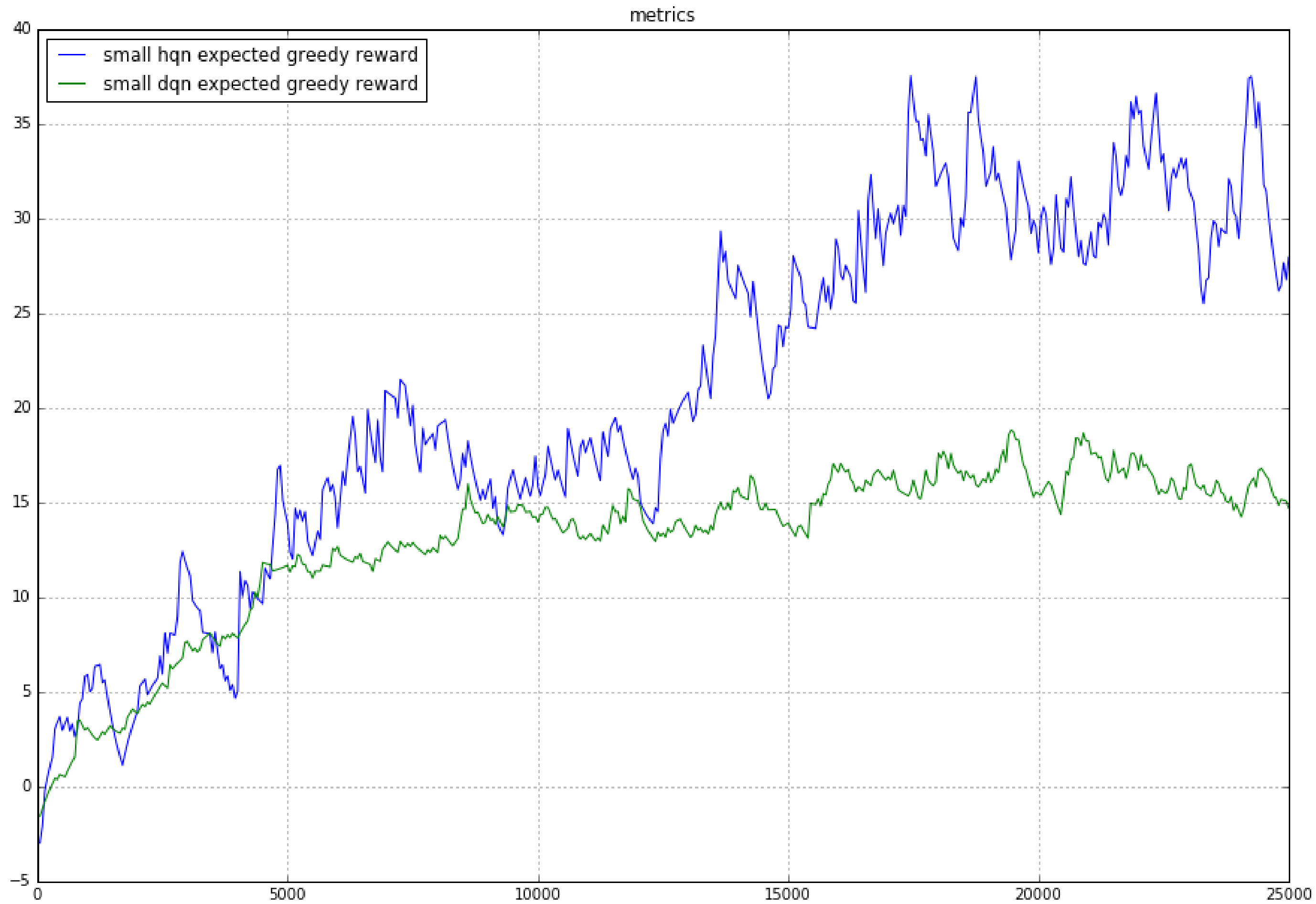
$$\pi(g|s, m_1) : argmax\, R$$

$$R_i = V_{A1}(s|m_1)$$

$$\pi(a|s, m_0) : R_i \rightarrow max$$

Learning curves





What's left behind

- Learning from human experience
- Inverse RL – learn **r** given near-optimal policy
- Model-based methods
- Whatever new stuff comes out '17

Course outro

This is almost the end...

Probably not too late to send homeworks.

Gonna be binge-checking on 19-20

Please tell us how to improve the course
http://bit.ly/2qwZSwN

Course outro