Hierarchical RL: Temporal Abstraction

Li, Ziyao; June 2nd, 2020

北京大學
PEKING UNIVERSITY

CONTENTS

01 Temporal Abstraction

R. S. Sutton *et al*: Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artif. Intell*. 112(1-2): 181-211 (1999)

Motivation.

北京大學
PEKING UNIVERSITY

To *abstract* a series of (primitive) actions into an *option* (Sutton et al, 1999).

Why:

-- to introduce temporal knowledge (abstraction) in RL (Sutton et al, 1999) ;

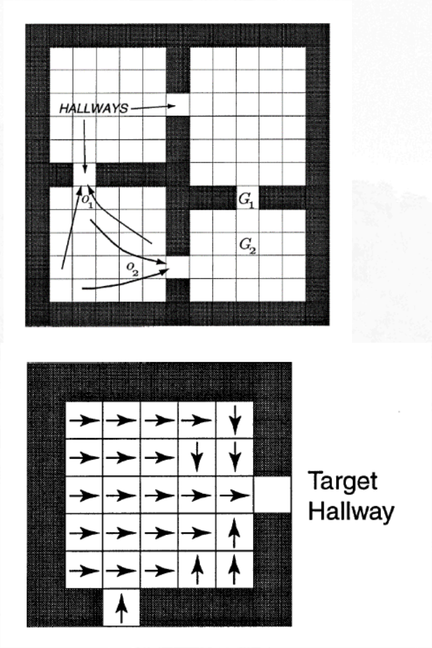
-- to address the sparsity of rewards (h-DQN, 2017) ;

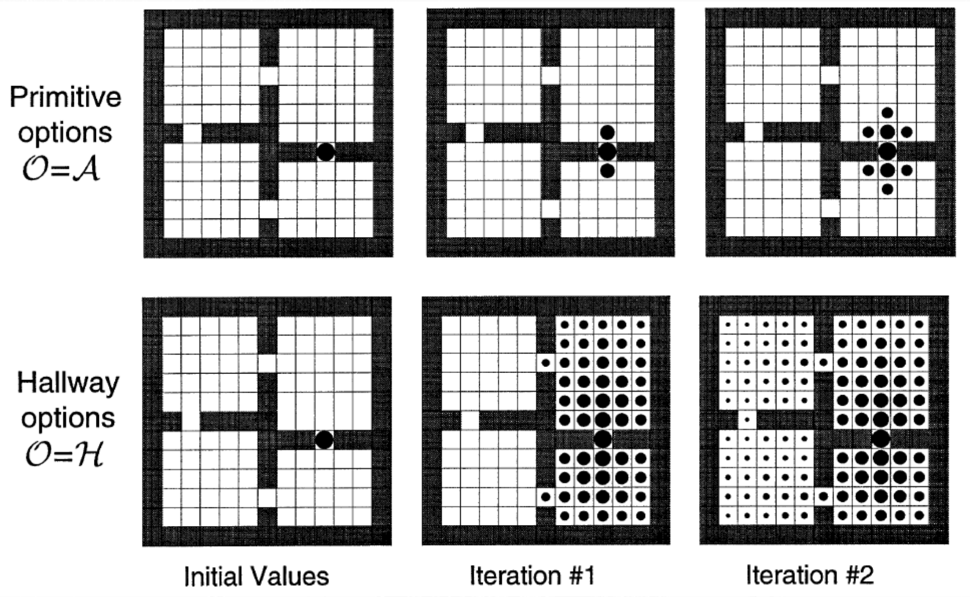
-- temporal resolution (Dayan & Hinton, 1993) ;

-- to improve sample efficiency (MLSH, 2019) ;

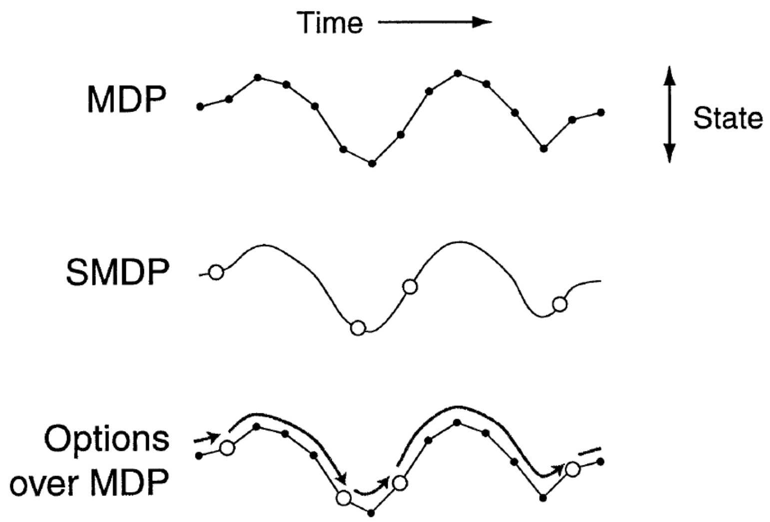
-- to behave more like human …

Some examples.





Options.



Definition of (Markov) options:

$$o = \langle I, \beta, \pi \rangle$$

$I \subset S$: a set of "entries".

$\beta: S^+ \to [0,1]$: probabilistic "exits"

$\pi: S \times A \to [0, 1]$: policy

(transition probability)

Decisions with options.

Available options: $O_s = \{o: s \in I_o\}, O = \bigcup_{s \in S} O_s$

Adapting actions as options: $o_a = \langle I_a, \beta(s) = 1, \pi(s, a) = 1 \rangle$

Markov (option) policy: $\mu: O \times A \to [0, 1]$

Flat policy: $\pi = flat(\mu)$

Action-Value fn. with options

北京大學 PEKING UNIVERSITY

Define

$$r_s^o = E\left(r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{k-1} r_{t+k} \mid o, s, t\right)$$

$$p_{s,s'}^o = \sum_{k=1}^{\infty} p(s', k)\gamma^k$$

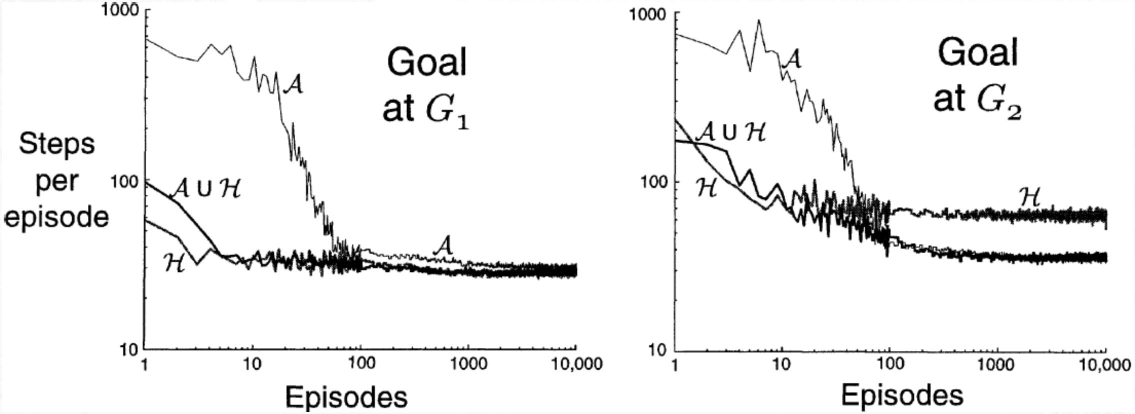
Then

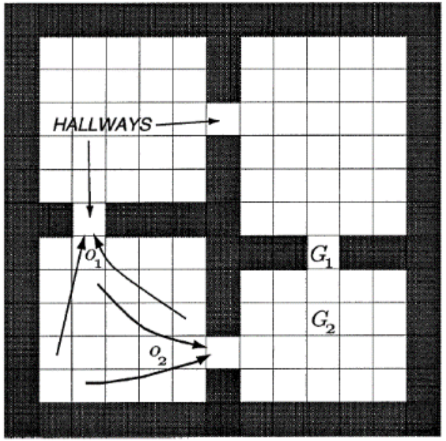
$$Q_O^*(s,o) = r_s^o + E_{s'}\left(\gamma^K \max_{o' \in O_{s'}} Q_O^*(s', o') \mid o, s\right)$$

(Recap in MDP)

$$Q^*(s,a) = r_s^a + E_{s'}\left(\gamma \max_{a' \in A_{s'}} Q^*(s', a') \mid a, s\right)$$

Experiments.





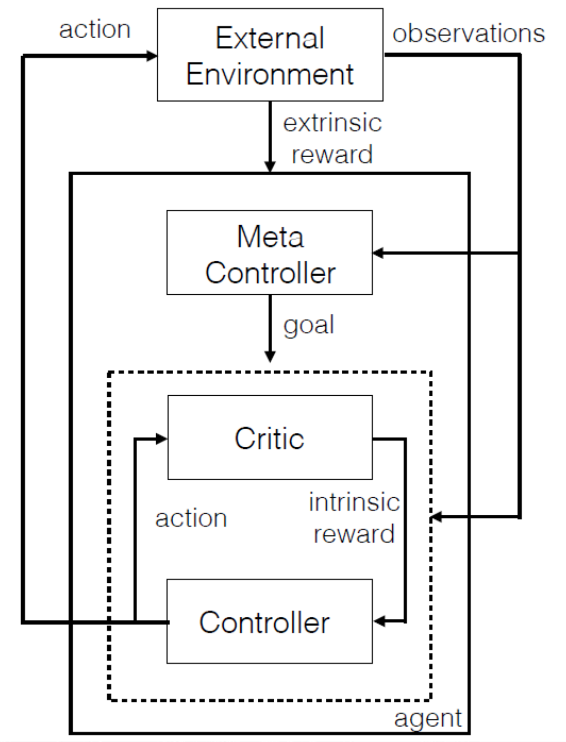
02 H-DQN

T. D. Kulkarni *et al*: Hierarchical Deep Reinforcement Learning:
Integrating Temporal Abstraction and Intrinsic Motivation. *NIPS 2016*: 3675-3683

Motivation.

Implementing options with DQN *(meta-controller & controller).*

- *meta-controller* chooses options with "goals", and pass it to the *controller.*



Corresponding Q-fn.

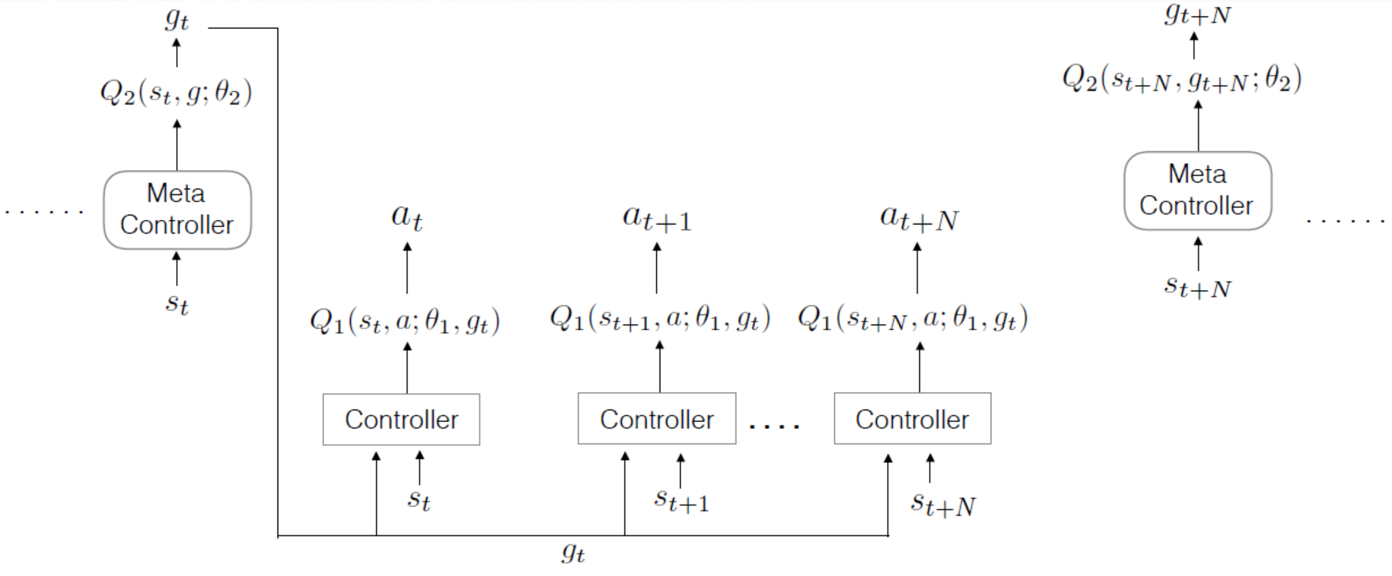
meta-controller

$$Q_2^*(s,g) = \max_{\pi_g} \mathrm{E}[\sum_{t'=t}^{t+N} f_{t'} + \gamma \, \max_{g'} Q_2^*(s_{t+N}, g') \mid s_t = s, g_t = g, \pi_g]$$

controller

$$Q_1^*(s,a;g) = \max_{\pi_{ag}} \mathrm{E}[r_t + \gamma \, \max_{a_{t+1}} Q_1^*(s_{t+1}, a_{t+1}; g) \mid s_t = s, a_t = a, g_t = g, \pi_{ag}]$$

Model Arch.

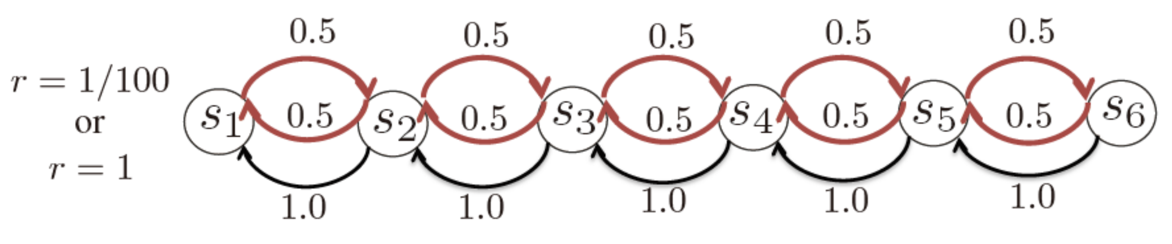


g_t
$Q_2(s_t, g; \theta_2)$
Meta Controller
s_t
a_t
a_{t+1}
a_{t+N}
$Q_1(s_t, a; \theta_1, g_t)$
$Q_1(s_{t+1}, a; \theta_1, g_t)$
$Q_1(s_{t+N}, a; \theta_1, g_t)$
Controller
Controller
Controller
s_t
s_{t+1}
s_{t+N}
g_t
g_{t+N}
$Q_2(s_{t+N}, g_{t+N}; \theta_2)$
Meta Controller
s_{t+N}

Learning algorithm.

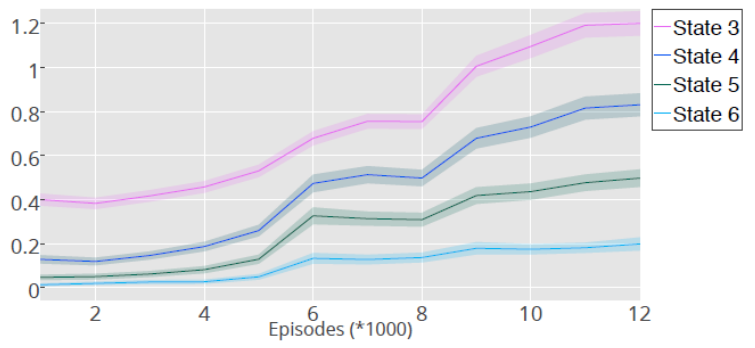
$\boxed{g \leftarrow \text{EPSGREEDY}(s, \mathcal{G}, \epsilon_2, Q_2)}$
while s is not terminal **do**
$F \leftarrow 0$
$s_0 \leftarrow s$
while **not** (s is terminal **or** goal g reached) **do**
$\boxed{a \leftarrow \text{EPSGREEDY}(\{s, g\}, \mathcal{A}, \epsilon_{1,g}, Q_1)}$
Execute a and obtain next state s' and extrinsic reward f from environment
Obtain intrinsic reward $r(s, a, s')$ from internal critic
$\boxed{\text{Store transition } (\{s, g\}, a, r, \{s', g\}) \text{ in } \mathcal{D}_1}$
$\text{UPDATEPARAMS}(\mathcal{L}_1(\theta_{1,i}), \mathcal{D}_1)$
$\text{UPDATEPARAMS}(\mathcal{L}_2(\theta_{2,i}), \mathcal{D}_2)$
$F \leftarrow F + f$
$s \leftarrow s'$
end while
$\boxed{\text{Store transition } (s_0, g, F, s') \text{ in } \mathcal{D}_2}$
if s is not terminal **then**
$g \leftarrow \text{EPSGREEDY}(s, \mathcal{G}, \epsilon_2, Q_2)$
end if
end while

Toy problem.



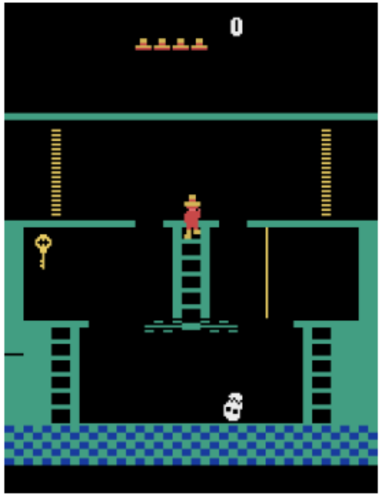
$r = 1/100$
or
$r = 1$
s_1
s_2
s_3
s_4
s_5
s_6
0.5
0.5
1.0

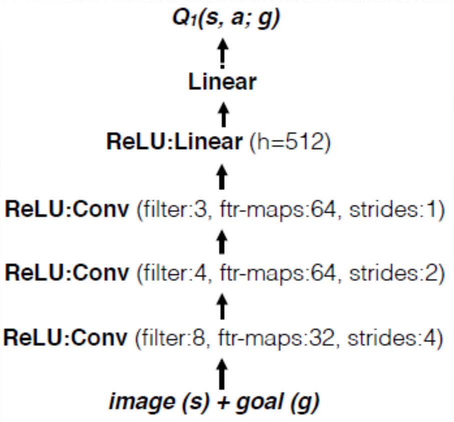
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
50
100
150
200
Steps
Baseline
Our Approach

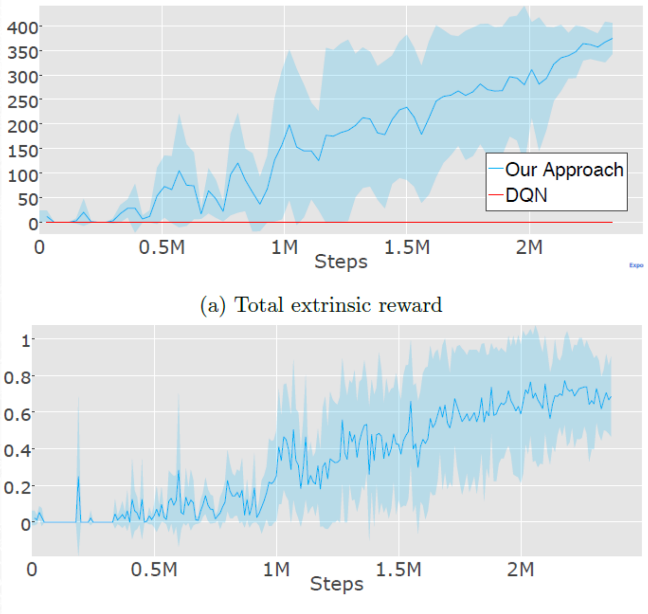


1.2
1
0.8
0.6
0.4
0.2
0
2
4
6
8
10
12
Episodes (*1000)
State 3
State 4
State 5
State 6

Sample Architecture:
for Montezuma's Revenge.







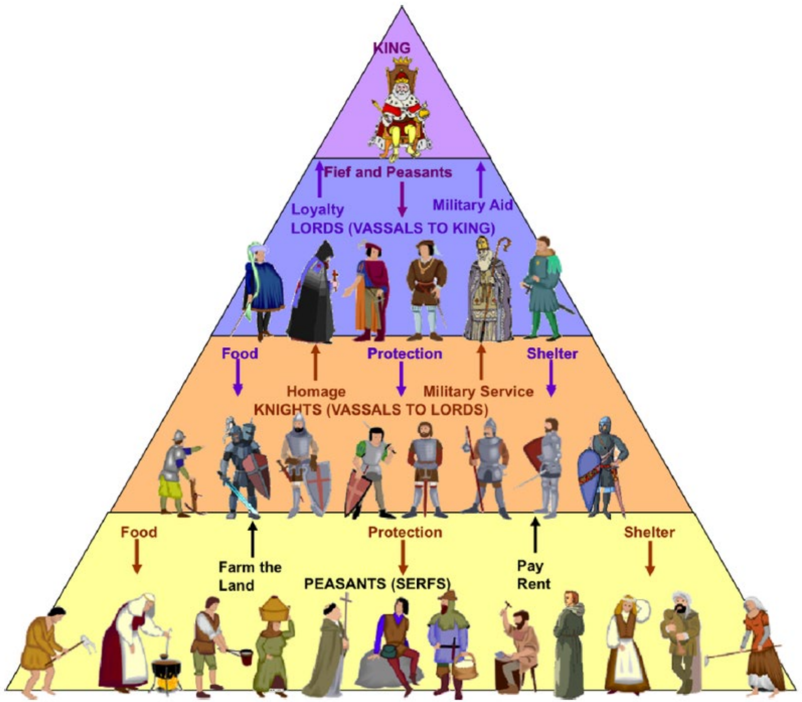
(a) Total extrinsic reward

(b) Success ratio for reaching the goal 'key'

03 Feudal (封建) Networks

P. Dayan, G. E. Hinton: Feudal Reinforcement Learning. *NIPS 1992*: 271-278
A. S. Vezhnevets *et al*: FeUdal Networks for Hierarchical Reinforcement Learning. *ICML 2017*: 3540-3549

Motivation.

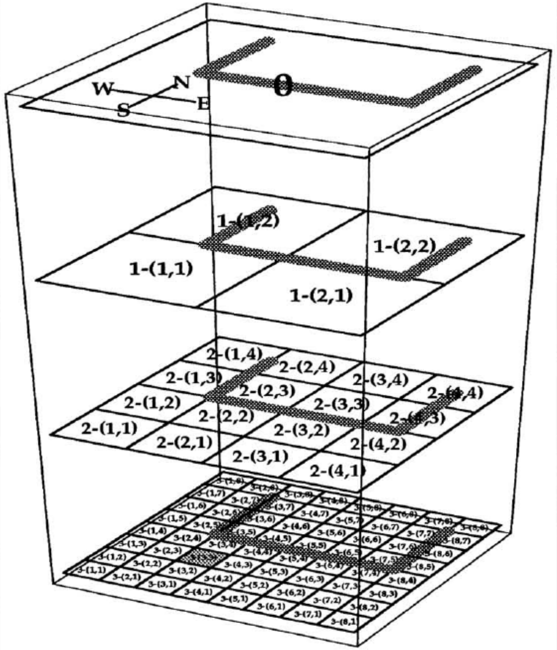


Feudal hierarchy. Source

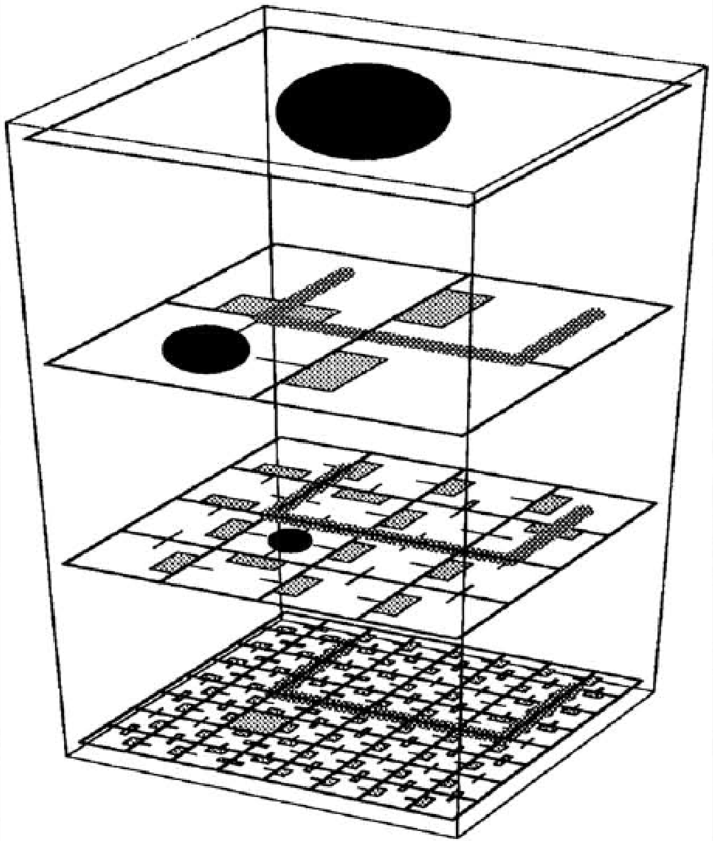
Imitating a (feudal) society where:

- *managers* assign tasks for *sub-managers* (*workers*) (Dayan & Hinton, 1992) .
- Different social roles have different perceptions (w.r.t. granularity)

Example.



W
N
S
E
O
1-(1,2)
1-(2,2)
1-(1,1)
1-(2,1)
2-(1,4)
2-(2,4)
2-(1,3)
2-(3,4)
2-(2,3)
2-(1,2)
2-(3,3)
2-(2,2)
2-(1,1)
2-(3,2)
2-(4,2)
2-(2,1)
2-(3,1)
2-(4,1)



Feudal Networks.

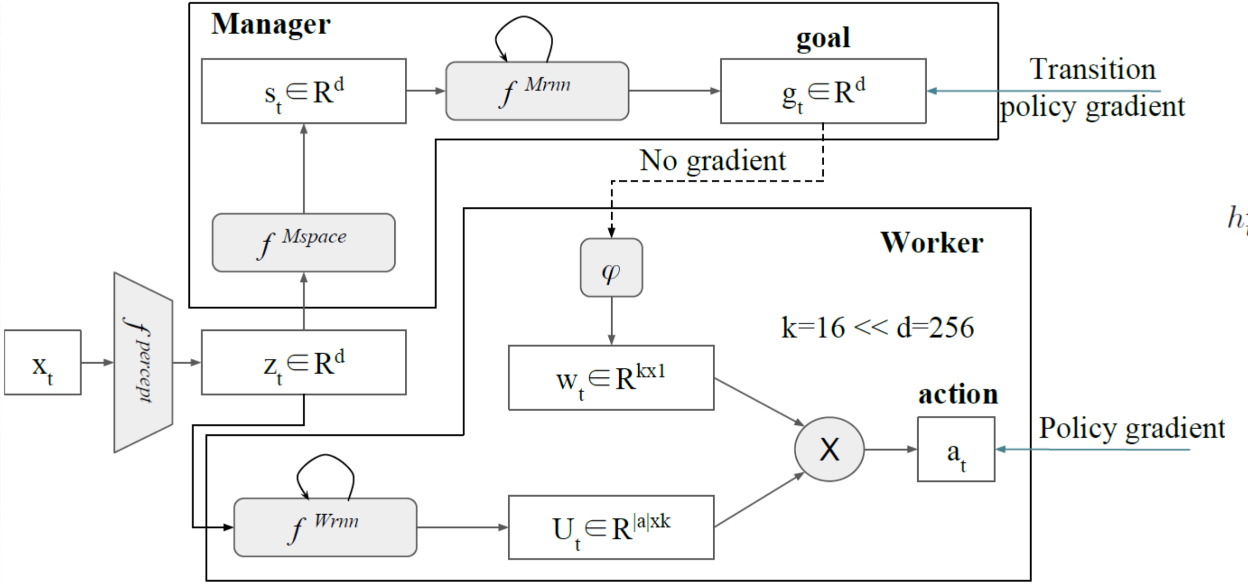
- This work is more like a fixed-length option instead of feudal learning.
- *The manager* announce a "goal" for the worker to achieve.
- In FUN, the authors required the "goal" to be explicitly meaningful (which is announced as the major difference), and the "goal" is defined to control the change of the state, *i.e.*

$$goal_t \sim \Delta s = s_{t+c} - s_t,$$

- where c is a hyper-parameter.

Architecture

北京大學 PEKING UNIVERSITY



$$z_t = f^{percept}(x_t)$$

$$s_t = f^{Mspace}(z_t)$$

$$h_t^M, \hat{g}_t = f^{Mrnn}(s_t, h_{t-1}^M); g_t = \hat{g}_t / ||\hat{g}_t||;$$

$$w_t = \phi(\sum_{i=t-c}^{t} g_i)$$

$$h^W, U_t = f^{Wrnn}(z_t, h_{t-1}^W)$$

$$\pi_t = SoftMax(U_t w_t)$$

Update.

北京大学 PEKING UNIVERSITY

Manager

Update: $$\nabla g_t = A_t^M \nabla_\theta d_{\cos}(s_{t+c} - s_t, g_t(\theta)),$$

(derived from): $$\nabla_\theta \pi_t^{TP} = \mathbb{E}\left[(R_t - V(s_t))\nabla_\theta \log p(s_{t+c}|s_t, \mu(s_t, \theta))\right]$$

Extr. Reward: $$A_t^M = R_t - V_t^M(x_t, \theta)$$

Worker

Update: $$\nabla \pi_t = A_t^D \nabla_\theta \log \pi(a_t|x_t; \theta)$$

Intr. Reward: $$r_t^I = 1/c \sum_{i=1}^{c} d_{\cos}(s_t - s_{t-i}, g_{t-i})$$

$$A_t^D = (R_t + \alpha R_t^I - V_t^D(x_t; \theta))$$

04 HRL as Meta Learning

Kevin Frans *et al*: Meta Learning Shared Hierarchies. *ICLR (Poster) 2018*

Motivation.

Meta Learning: from a task to a distribution of tasks.

Different tasks share a set of sub-policies (motor primitives).

Formulation:

- P_M: a distribution of MDPs on the same state & action space (S, A).
- $\pi_{\phi,\theta}(a|s)$: the policy with ϕ shared across tasks, θ task specific.
- Target: $\max_{\phi} E_M[R], \;\; R = r_0 + r_1 + \cdots + r_{T-1}$

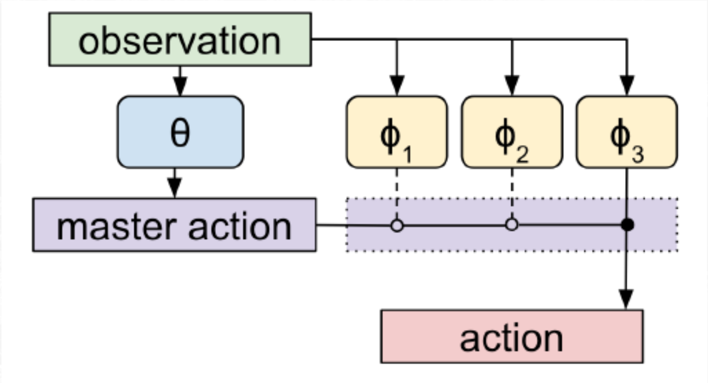
Meta learning with HRL.

Θ chooses among $\phi = \{\phi_1, \cdots, \phi_k\}$, where ϕ_k defines $\pi_{\phi_k}(a|s)$

Θ functions each N steps (fixed interval *options*)

Training:

- warm-up (θ) + joint training (θ, ϕ)

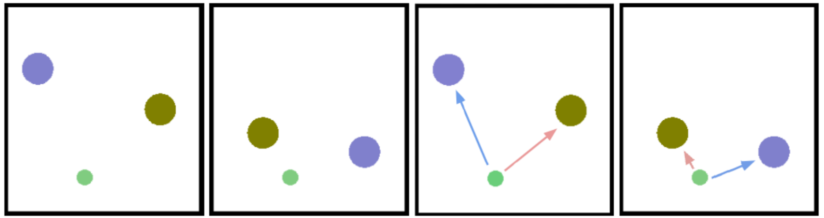


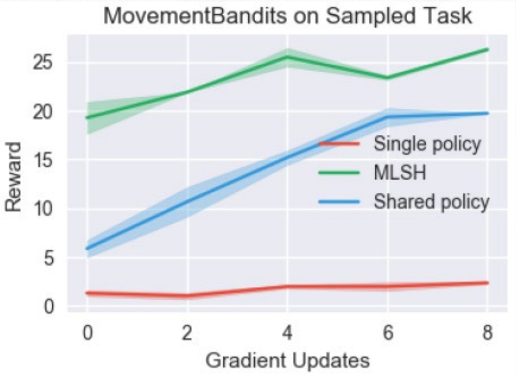
Training scheme.

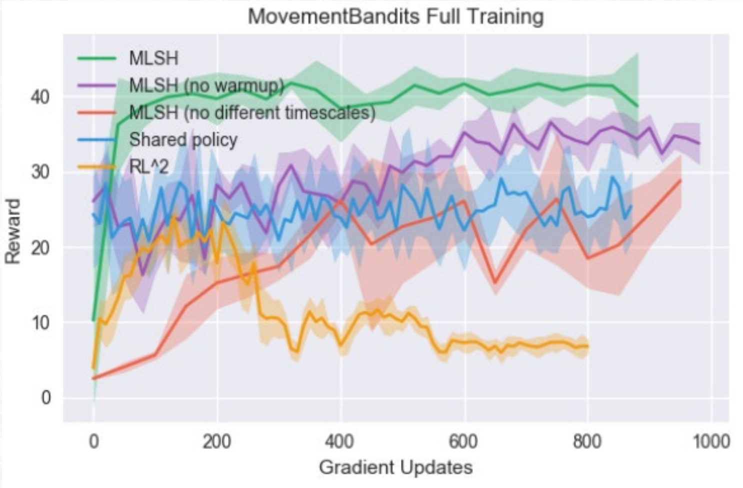
Algorithm 1 Meta Learning Shared Hierarchies

Initialize ϕ
repeat
  Initialize θ
  Sample task $M \sim P_M$
  for $w = 0, 1, ...W$ (warmup period) **do**
    Collect D timesteps of experience using $\pi_{\phi,\theta}$
    Update θ to maximize expected return from $1/N$ timescale viewpoint
  end for
  for $u = 0, 1, ...U$ (joint update period) **do**
    Collect D timesteps of experience using $\pi_{\phi,\theta}$
    Update θ to maximize expected return from $1/N$ timescale viewpoint
    Update ϕ to maximize expected return from full timescale viewpoint Only for the activated sub-policies
  end for
until convergence

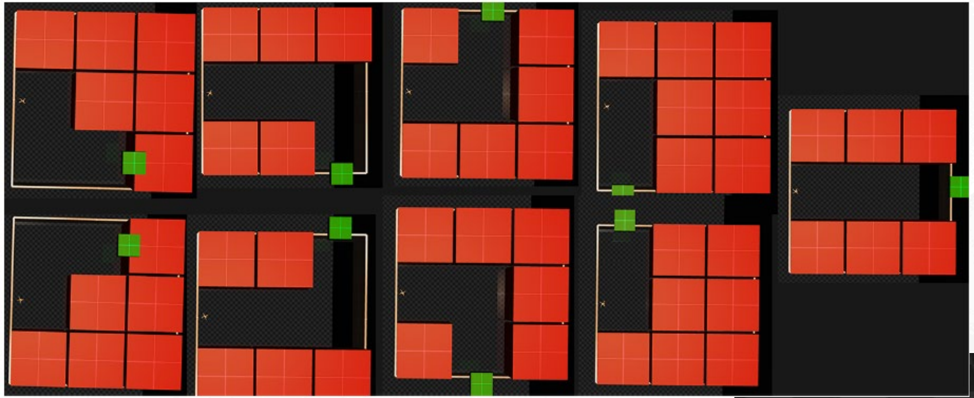
Some results: 2D bandits

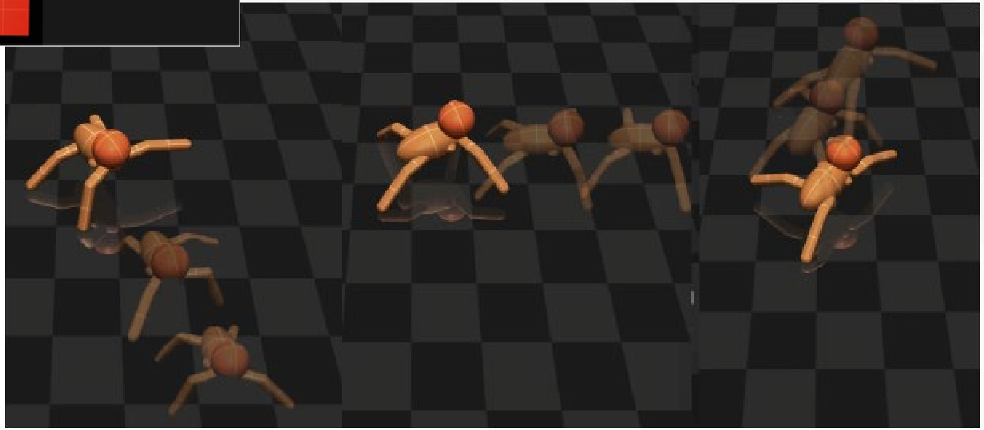






Some results: ant walk





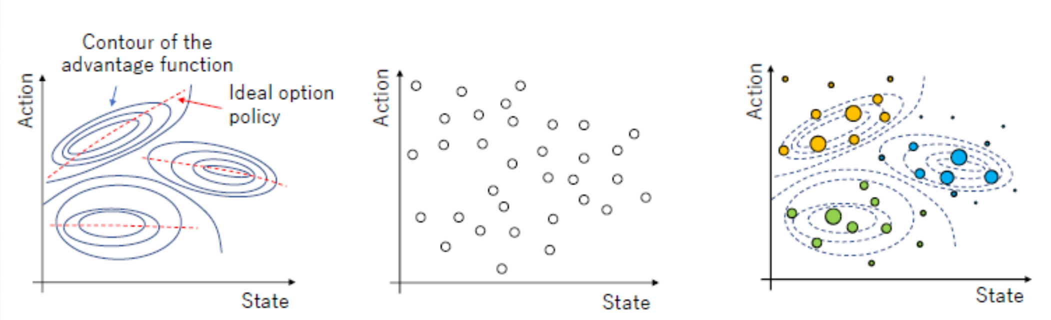
05 HRL with Infomax

Takayuki Osa *et al*: Hierarchical Reinforcement Learning via
Advantage-Weighted Information Maximization. *ICLR (Poster) 2019*

Motivation.

- Latent variable *o* in HRL can be defined differently, even when the step-wise policies are the same.
- AdInfoHRL aims to fine a "best" (tbd) hierarchical structure that improves sample-efficiency most.
- The "bestness" is defined via Infomax (mutual information maximization).

- Indeed, AdInfoHRL implements a clustering process in the (s,a) space with RIM.



Preliminaries (RIM).

Considering a supervised task with (x, y), the regularized infomax (RIM) first builds a conditional model w.r.t. η, $\hat{p}(y|x;\eta)$, and then minimizes

$$l(\eta) - \lambda I_\eta(x,y),$$

where

$$I_\eta(x,y) = H(y) - H(y|x;\eta).$$

$l(\eta)$ is the regularization and $I_\eta(x,y)$ is the mutual information (MI).

Preliminaries (HRL).

北京大学 PEKING UNIVERSITY

Let $d^\pi(\boldsymbol{s}) = \sum_{t=0}^{T} \gamma^t p(\boldsymbol{s}_t = \boldsymbol{s})$

Then the object of RL is:

$$J(\pi) = \iint d^\pi(\boldsymbol{s})\pi(\boldsymbol{a}|\boldsymbol{s})Q^\pi(\boldsymbol{s},\boldsymbol{a})\mathrm{d}\boldsymbol{a}\mathrm{d}\boldsymbol{s}$$

The object of HRL is:

$$J(\pi) = \iint d^\pi(\boldsymbol{s}) \sum_{o \in \mathcal{O}} \pi(o|\boldsymbol{s})\pi(\boldsymbol{a}|\boldsymbol{s},o)Q^\pi(\boldsymbol{s},\boldsymbol{a})\mathrm{d}\boldsymbol{a}\mathrm{d}\boldsymbol{s}$$

AdInfoHRL

北京大学 PEKING UNIVERSITY

- The policy is defined as $\pi_{ad}(a|s) = softmax\left(A^{\pi}(s,a)\right)$,
- where $A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$ is the *advantage function*.
- The option o is defined as a (discrete) latent variable (clusters of (s,a)) with conditional probability $\hat{p}(o|s,a;\eta)$.
- Correspondingly, the loss

$$L_{opt} = l(\eta) - \lambda I(o,(s,a);\eta)$$

where

$$I(o,(s,a);\eta) = H(o;\eta) - H(o|s,a;\eta)$$

AdInfoHRL

- The policy is defined as $\pi_{ad}(a|s) = softmax(A^\pi(s,a))$,
- where $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$ is the *advantage function*.
- The option o is defined as a (discrete) latent variable (clusters of (s,a)) with conditional probability $\hat{p}(o|s,a;\eta)$.
- Correspondingly, the loss

$$L_{opt} = l(\eta) - \lambda I(o,(s,a);\eta)$$

where

$$I(o,(s,a);\eta) = H(o;\eta) - H(o|s,a;\eta)$$

and $l(\eta) = KL(p(o|\tilde{s},\tilde{a})||p(o|s,a))$ is modeled via *virtual adversarial training* (VAT)

Maths.

北京大学 PEKING UNIVERSITY

- Addressing $H(o;\eta)$ and $H(o|s,a;\eta)$:

$$p(o) = E_{(s,a)\sim\pi_{ad}}[p(o|s,a;\eta)]$$

$$H(o|s,a;\eta) = E_{(s,a)\sim\pi_{ad}}[p(o|s,a;\eta)\log p(o|s,a;\eta)]$$

- where the density of $(s,a)\sim\pi_{ad}$ must be modeled.
- AdInfoHRL: advantage-weighted importance.

Advantage-weighted importance.

北京大學 PEKING UNIVERSITY

- $\hat{p}(o) = \frac{1}{N}\sum_{i=1}^{N} \widetilde{W}(s_i, a_i) p(o|s_i, a_i; \eta)$
- $\widehat{H}(o|s, a) = \frac{1}{N}\sum_{i=1}^{N} \widetilde{W}(s_i, a_i) p(o|s_i, a_i; \eta) \log p(o|s_i, a_i; \eta)$
- Sketch: to address the ***difference of the induced distributions*** of the true policy π_{ad} and the behavior policy β.
- Let $W(s, a) = \frac{\pi_{ad}(s,a)}{\beta(s,a)}$ and $\widetilde{W}(s, a) = \frac{W(s,a)}{\sum W(s,a)} = \frac{\exp(A(s,a))}{\beta(s,a)} \Big/ \sum \frac{\exp(A(s,a))}{\beta(s,a)}$
- $p(o) = \int p_\beta\,(s, a)\, \frac{p_\pi(s,a)}{p_\beta\,(s,a)}\, p(o|s, a; \eta)\, dads = E_\beta[W(s, a) p(o|s, a; \eta)]$

2020

Thanks

Presented by Li, Ziyao

北京大學
PEKING UNIVERSITY