Deep Hierarchical Reinforcement Learning Algorithms in Partially Observable Markov Decision Processes

Ph.D. Dissertation Defense

Presenter: Le Pham Tuyen
Advisor: Prof. TaeChoong Chung, Ph.D.
Artificial Intelligence Laboratory
Department of Computer Science and Engineering

Kyung Hee University, 14^{th} November 2018

Thesis Outline

경희대학교
KYUNG HEE UNIVERSITY

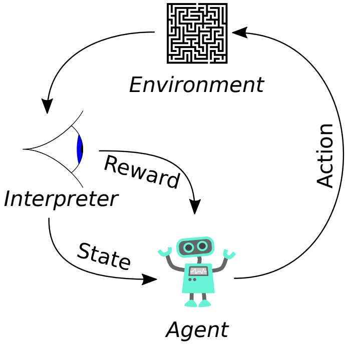
1. Introduction
2. Challenges
3. Thesis Contributions
4. Background and Related Work
5. Proposed Methodologies
6. Experiments and Results
7. Conclusion and Future Work
8. References

Introduction

Reinforcement Learning

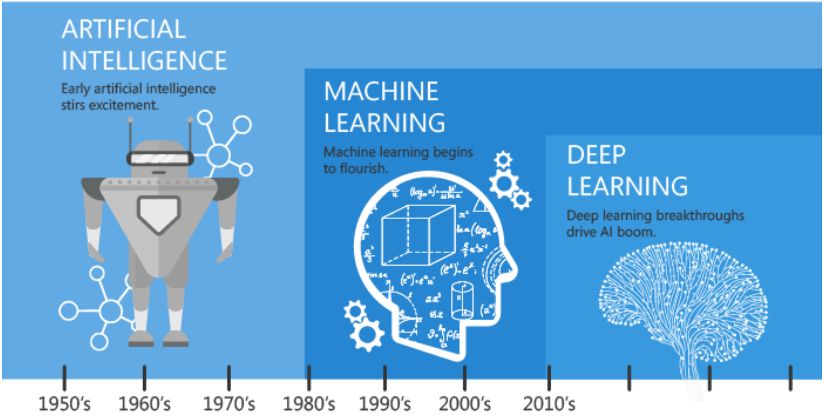
Reinforcement Learning

An area of **Machine Learning** *concerned with how software agents take actions in an environment so as to maximize cumulative reward.*



Machine Learning

- We can answer the 4 major questions:
  - How much/How many?
  - Which category?
  - Which group?
  - Which action?



How much / How many?

- What will be the temperature tomorrow?
- What will be my energy costs next week?
- How many new user will visit next month?

⇒ Regression

Which category?

- Is there a cat or a dog on the image?
- Which emails are spam emails?
- What is the category of this news article (finance, weather, entertainment, sport, . . .)?

⇒ Classification

Which group?

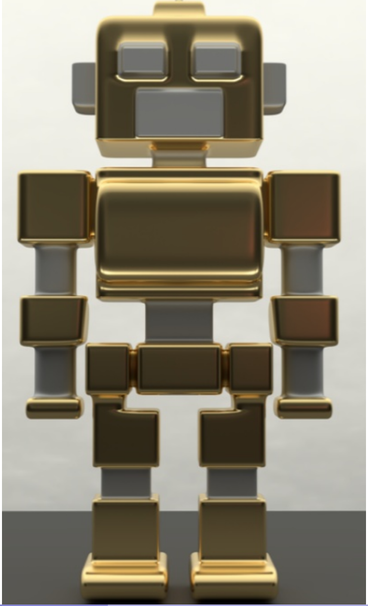
- Which customers have the same favorite product?
- Which visitors like the same movie?
- Which documents has the same topic?

$\Rightarrow$ Clustering

Which action?

- Should I rise or lower the temperature?
- Should I break or accelerate?
- What is the next move for this Go match?

⇒ Reinforcement Learning (RL)



RL application areas

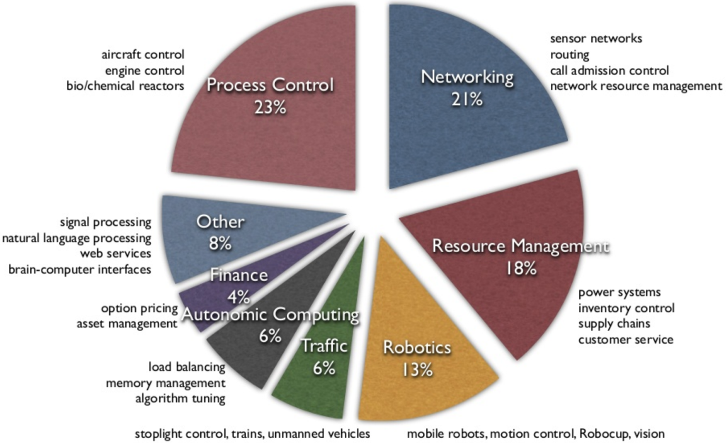


Figure: Rich Sutton. Deconstructing Reinforcement Learning. ICML 09

Era of Deep Reinforcement Learning

경희대학교
KYUNG HEE UNIVERSITY

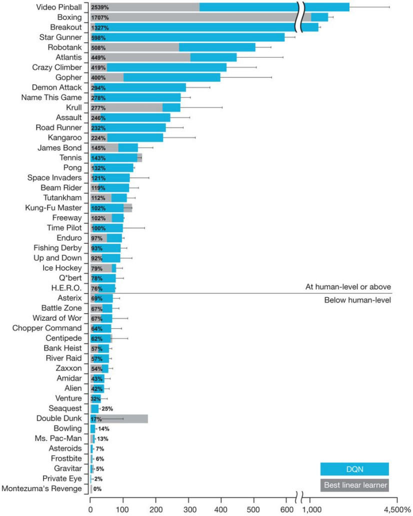


Figure: DQN in Atari Games

(a) Go game

(b) Starcraft

(c) DotA

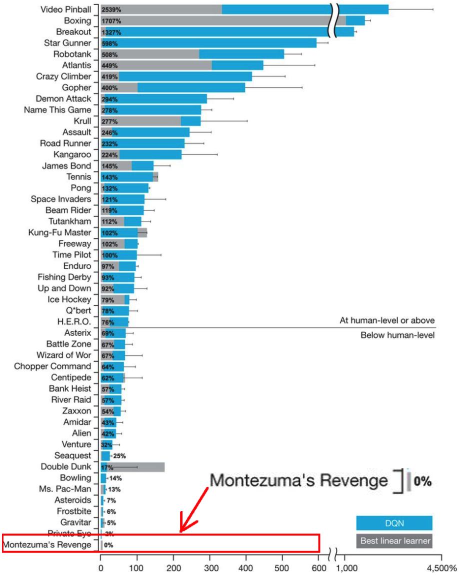
(d) Poker

Figure: Domains which the agent defeats human

Challenges

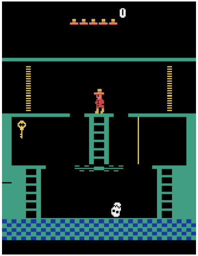
Challenge 1

경희대학교
KYUNG HEE UNIVERSITY



Hierarchical Task

DQN as well as plain DRL algorithms fails to solve the task having multiple subtasks (hierarchical task) such as Montezuma's Revenge in Atari Game 2600



Montezuma's Revenge Game

Challenge 2

Partial Observability

- *Most of studies assume that an agent can observe the environment states fully (***MDP***)*
- *However, it does not reflect the nature of real-world applications, where the agent only observes a partial states (***POMDP***)*

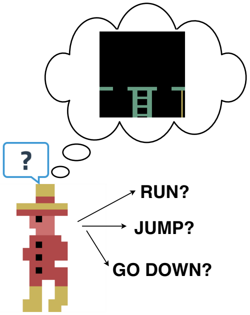
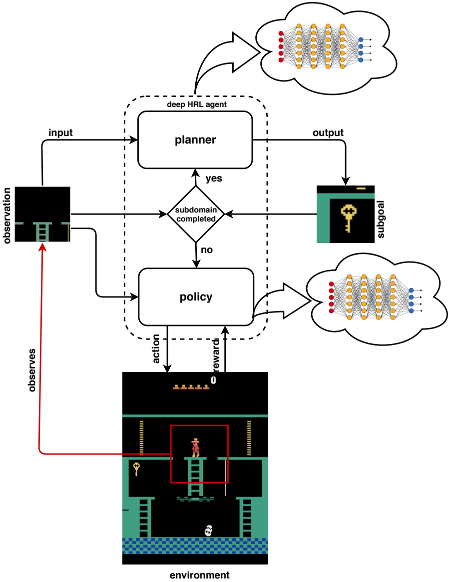


Figure: The agent takes the action under partial observability

Proposed Concept

경희대학교
KYUNG HEE UNIVERSITY



We want to propose a deep HRL algorithm for solving hierarchical tasks under partial observability

- The proposed frameworks employ deep neural network as policies.
- The proposed frameworks use limited observations to make decisions.
- The proposed frameworks can solve hierarchical tasks

Thesis Contributions

Thesis Contributions

- **Develop:** **h**ierarchical **D**eep **R**ecurrent **Q**-Learning algorithms (**hDRQNs**) in order to handle **hierarchical tasks** in **POMDP**. Particularly,
  - We develop hDRQNv1 algorithm which learns a framework of hierarchical polices.
    - ★ Two levels of hierarchical polices: meta-controller is the upper policy and sub-controller is the lower policy.
    - ★ Two hierarchical policies integrated recurrent neural networks are expected to overcome the challenges under partial observability
  - We develop hDRQNv2 algorithm of a proposed framework which integrates recurrent neural networks in a different way, thus expected to have better performance.
- To the best of our knowledge, our research is the first study that learns Montezuma's Revenge under partial observability.

Background and Related Work

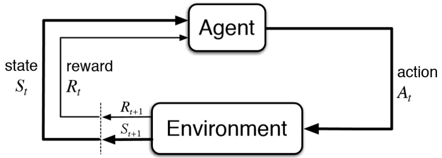
Background and Related Work

- Reinforcement learning (Markov Decision Process)
- Hierarchical reinforcement learning (Semi Markov Decision Process)
- Planing under partial observability (Partial Observation Markov Decision Process)
- Related works:
  - Deep Q Networks (DQN)
  - Deep Recurrent Q Network (DRQN)
  - Hierarchical Deep Q Network (hDQN)

Markov Decision Process (MDP)

경희대학교 KYUNG HEE UNIVERSITY

- RL can be formalized as a MDP with five elements $\{\mathcal{S}; \mathcal{A}; r; \mathcal{P}; \gamma\}$



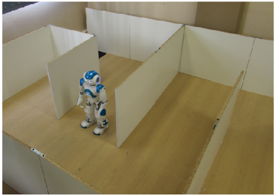
  - $\mathcal{S}$ state space
  - $\mathcal{A}$ action space
  - $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ reward function
  - $\mathcal{P}(s'|s, a)$ transition dynamics
  - γ discount factor
- Markov property: $\mathcal{P}(s_{t+1}|s_1, a_1, \ldots, s_t, a_t) = \mathcal{P}(s_{t+1}|s_t, a_t)$
- A policy π is a map from state to action. E.g.
  - Deterministic policy: $a = \pi(s)$
  - Stochastic policy: $\pi(a|s) = P[a_t = a|s_t = s]$

Goal of RL

Find an optimal policy π^* in order to maximize the expected discounted reward: $J(\pi) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r(a_t, s_t)\right]$

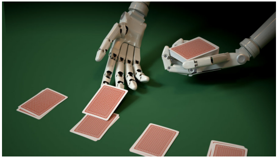
Partial Observation Markov Decision Process (POMDP)

- Agent observes the entire environment → **MDP**
- Agent only observes a part of environment → **POMDP**
- **POMDP** is popular in the real-world applications. E.g.
  - A robot with camera vision isn't told its absolute location
  - A trading agent only observes current prices
  - A poker playing agent only observes public cards



(a) Robot Navigation

(b) Trading Bot



(c) Poker Bot

Some POMDP domains

Partial Observation Markov Decision Process (POMDP)

경희대학교 KYUNG HEE UNIVERSITY

- **POMDP** is defined as a tuple of six components $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \Omega, \mathcal{Z}\}$
  - $\mathcal{S}, \mathcal{A}, \mathcal{P}, r$ are the state space, action space, transition function and reward function, respectively, as in a **MDP**.
  - Ω and $\mathcal{Z}$ are the observation space and observation model, respectively
- The agent cannot observe the whole environment, thus, maintain a hidden state b called *belief state*

Definition

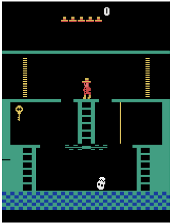
Belief state defines the probability of being in state s according to its history of actions and observations; and can be updated using the Bayes rule:

$$b'(s') \propto \mathcal{Z}(o|s', a) \sum_{s \in \mathcal{S}} \mathcal{P}(s'|s, a) b(s).$$

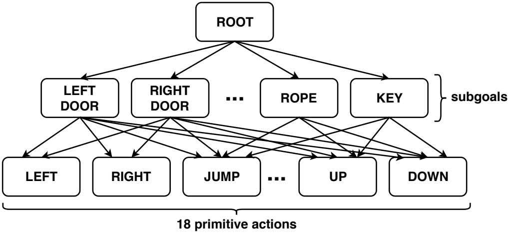
- Updating belief state require a high computational cost and expensive memory $\rightarrow$ take advantages of RNNs

Semi Markov Decision Process (SMDP)

- Hierarchical tasks are popular in real-world applications. E.g.
  - An agent navigates to the key before reaching the door to open.
  - Tasks of a taxi: go to to the passengers, pick up, go to to the destination, take off.
  - A robot plans to go to the door before going to the destination.



(a) Montezuma's Revenge



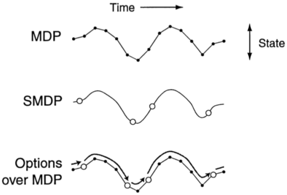
(b) The hierarchy of Montezuma's Revenge domain

Hierarchical Domain

- **SMDP** is an extensional theory of MDP, was developed to deal with challenges in hierarchical tasks.

SMDP = Options + MDP

- **SMDP** = Options over MDP.

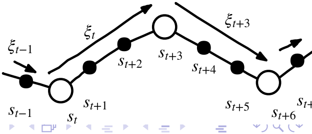


Definition

An option $\xi \in \Xi$ is defined by three elements:

- *An option's policy π,*
- *A termination condition β*
- *An initiation set $\mathcal{I} \subseteq \mathcal{S}$ denoted as the set of states in the option*

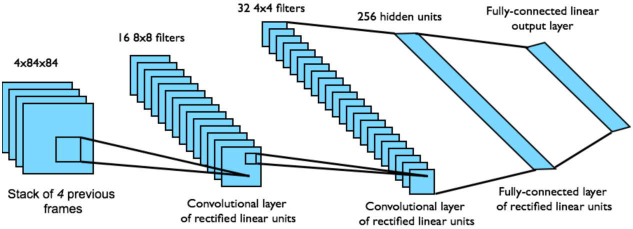
- A policy over options $\mu(\xi|s)$ is introduced to select options
- An option is executed as follows:
  - Under option ξ_t, state s_t, the action a_t is selected based on π
  - The environment transits to state s_{t+1}
  - The option executes until state s_{t+3}
  - The next option is selected $\xi_{t+3} = \mu(s_{t+3})$



Deep Reinforcement Learning (1)

경희대학교 KYUNG HEE UNIVERSITY

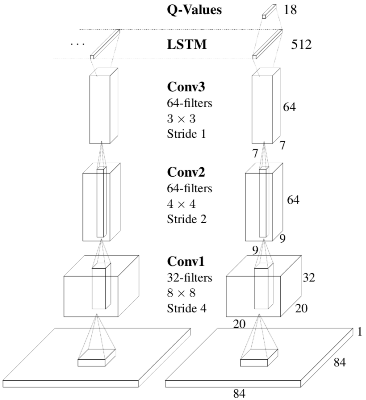
- Deep Q Learning (DQN) for Atari Games
  - End-to-end learning of values $Q(s, a)$ from raw pixels
  - Input state s is stack of raw pixels from last 4 frames
  - Output is $Q(s, a)$ for 18 joystick/button positions
  - Hidden layers are the combination of CONV, FC, ReLU
  - Stabilization techniques:
    - ★ Experience replay.
    - ★ Delayed target network.



- Other tricks:
  - Double Deep Q Learning (DDQN)
  - Dueling network
  - Prioritized replay

Deep Reinforcement Learning (2)

경희대학교 KYUNG HEE UNIVERSITY

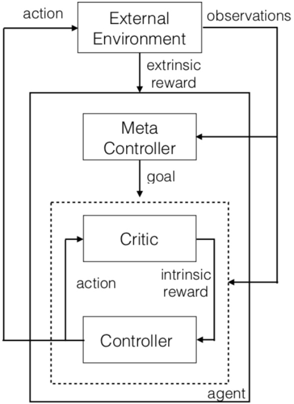


- Limitations of DQN and its derivations:
  - Only learning from a limited number of past states (last 4 frames)
  - Cannot deal with POMDP domains
- Deep Recurrent Q-Network(DRQN) [6]:
  - A combination of a Long Short Term Memory (LSTM) and a DQN
  - Better handles the loss of information (POMDP)

- Other tricks combining with DRQN [6]:
  - Updating DRQN techniques: *Bootstrapped Sequential Updates* vs *Bootstrapped Random Updates*
  - Ignore first observations in a sequence of transitions when updating the Q value function

Deep Reinforcement Learning (3)

경희대학교 KYUNG HEE UNIVERSITY



- hDQN framework [3]
  - Two levels of controllers: *meta controller* and *controller*
  - The *meta controller* produces a subgoal.
  - The *controller* performs primitive actions to obtain the subgoal.
  - The set of subgoals is predefined and fixed.
  - The *meta controller* and the *controller* are built from DQN networks
  - *Extrinsic* is reward of the meta controller and *intrinsic* is reward of the controller
  - Only deal with fully observable domains

- Others:
  - Option Critic framework [1] and Feudral framework [2]
  - Discovering subgoals [4]
  - Adaptively finding a number of options [5]

Proposed Methodologies

hDRQN: Key Terminologies (1)

Subdomain (ξ)

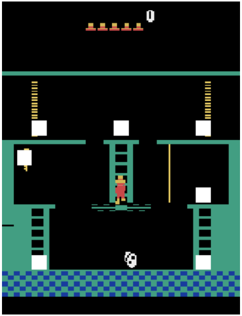
- *A domain = multiple subdomains.*
- *A subdomain $\Leftrightarrow$ an option ξ.*

E.g. Domain: Montezuma's Revenge.
Subdomains: move to the left door, move to the key, ...

Subgoal (g)

Each subdomain has a subgoal $g \in \Omega$

E.g. White rectangles (left image)



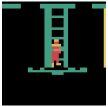


Figure: Montezuma's Revenge

Observation (o)

A partial of the environment ($o \in \Omega$) which the agent can observe

E.g. The pixels around the agent (right image)

hDRQN: Key Terminologies (2)

Meta-controller (META)

Equivalent to a "policy over subgoals" that receives the current observation o_t and determines the new subgoal g_t

- E.g. In Montezuma's Revenge, META is used to select new subgoal.

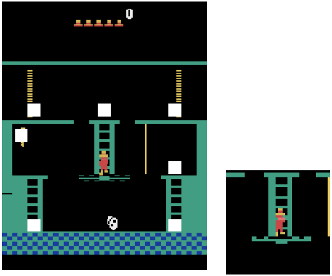


Figure: Montezuma's Revenge

Extrinsic Reward (r^{ex})

Use to evaluate the goodness of META.

- E.g. In Montezuma's Revenge, $r^{ex} = 1$ if agent obtains the key or opens the doors, otherwise 0

hDRQN: Key Terminologies (3)

Sub-controller (SUB)

Equivalent to the *option's policy*, *which directly interacts with the environment by performing action* a_t

- E.g. In Montezuma's Revenge, SUB controls the agent to move between subgoals.

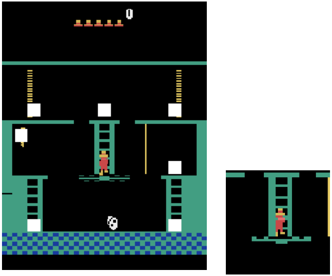


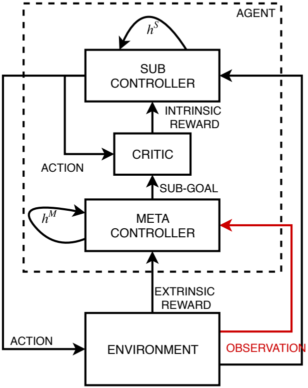
Figure: Montezuma's Revenge

Intrinsic Reward (r^{in})

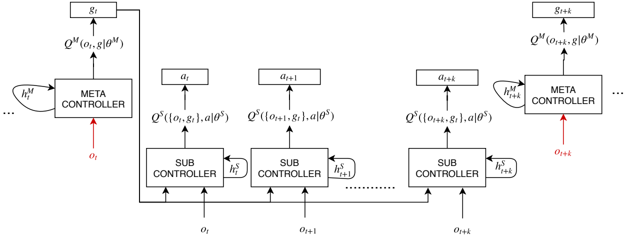
Use to evaluate the goodness of SUB.

- E.g. In Montezuma's Revenge, $r^{in} = 1$ if agent obtains the subgoal, otherwise 0

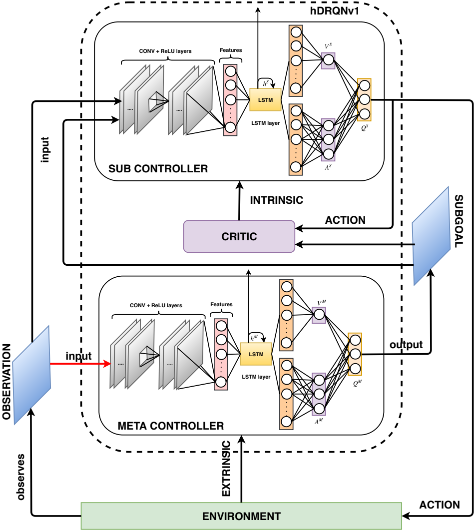
hDRQN: Framework 1



- hDRQNv1:
  - Inspired by hDQN framework [3]
  - Build on two deep *recurrent* neural policies.
  - Input is a single frame (hDQN uses 4 frames)



hDRQN: Framework 1 (Extended)



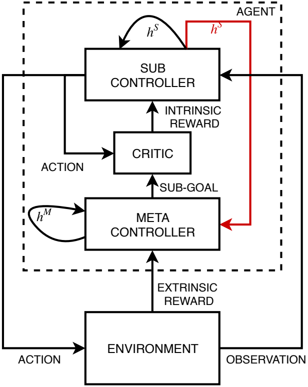
META:

- Input: Observation o
- Feature extraction: 4 CONV layers and ReLU layers.
- LSTM is integrated in front of the features.
- The output of LSTM is put into Dueling network ([7])
- Output: Q subgoal values $Q^M(o, g)$

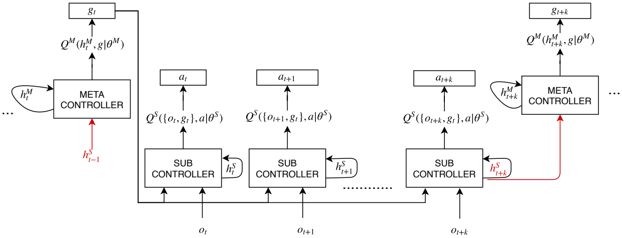
SUB:

- Input: Observation o and current subgoal (g)
- Other part: same as META
- Output: Q action values $Q^S(\{o, g\}, a)$

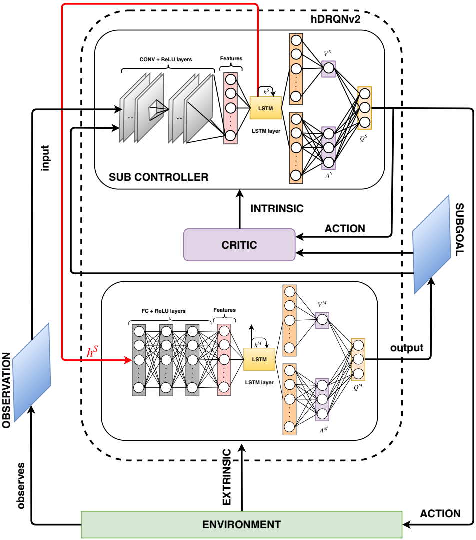
hDRQN: Framework 2



- hDRQNv2
  - An improved version of hDRQNv1
  - Input of META is the internal states of LSTM layer in SUB



hDRQN: Framework 2 (Extended)



META:

- Input: hidden states from SUB h^S
- Feature extraction: Three fully connected layers and ReLU layers.
- Other part has the same architecture as META of framework 1

SUB:

- Same architecture as SUB of framework 1

hDRQN: Q values

경희대학교
KYUNG HEE UNIVERSITY

- META Q subgoal values:

$$h_t^M, Q^M(o_t, g_t) = f^M(\Phi^M, h_{t-1}^M)$$

- SUB Q action values:

$$h_t^S, Q^S(\{o_t, g_t\}, a_t) = f^S(\Phi^S, h_{t-1}^S)$$

- Where:
  - f^M and f^S are the recurrent networks of the META and SUB.
  - h_t^M and h_t^S are internal states constructed by recurrent networks.
  - Φ^M and Φ^S are the features of META and SUB.

$$\Phi^M = \begin{cases} f^{extract}(o_t) & \text{framework 1} \\ f^{extract}(h^S) & \text{framework 2} \end{cases}$$

$$\Phi^S = f^{extract}(o_t, g_t)$$

  - $f^{extract}$ is neural networks to extract features from input (E.g. CONV, FC, ReLU, . . .)

hDRQN: Learning META

경희대학교 KYUNG HEE UNIVERSITY

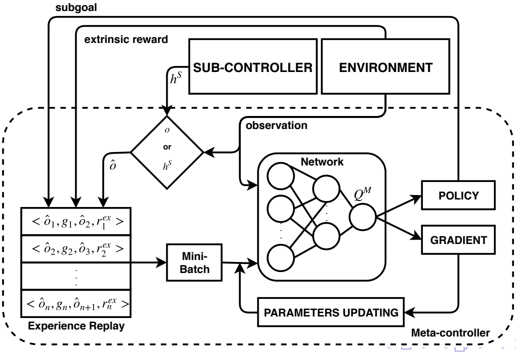
- Optimizing META by minimizing loss functions:

$$\mathcal{L}^M = \mathbb{E}_{(o,g,o',g',r^{ex})\sim\mathcal{M}^M}\left[y_i^M - \mathcal{Q}^M(o,g)\right]$$

- Where:
  - y_i^M is target values of META

$$y_i^M = r^{ex} + \gamma \mathcal{Q}^{M'}(o', \text{argmax}_{g'} \mathcal{Q}^M(o', g'))$$

- Minibatch Sampling Strategy: Bootstrapped Random Updates [6].



hDRQN: Learning SUB

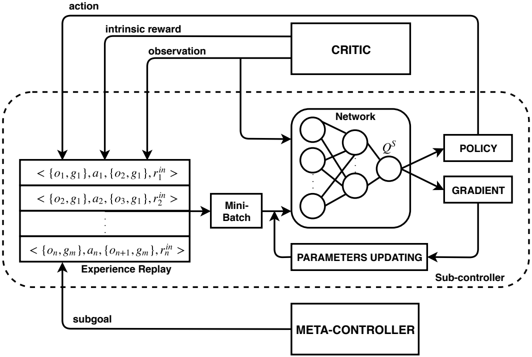
- Optimizing SUB by minimizing loss functions:

$$\mathcal{L}^S = \mathbb{E}_{(o,g,a,r^{in})\sim\mathcal{M}^S}\left[y_i^S - \mathcal{Q}^S(\{o,g\},a)\right]$$

- Where:
  - y_i^S are target values of SUB

$$y_i^S = r^{in} + \gamma \mathcal{Q}^{S'}(\{o',g\}, \mathrm{argmax}_{a'}\, \mathcal{Q}^S(\{o',g\},a'))$$

- Minibatch Sampling Strategy: Bootstrapped Random Updates [6].



hDRQN: Sampling Strategy

경희대학교 KYUNG HEE UNIVERSITY

- Bootstrapped Random Updates [6] is compatible with recurrent neural networks:
  - Randomly selects a batch of episodes from the experience replay
  - For each episode, we begin at a random transition and select a sequence of n transitions
  - For each controller, we have n^M (META) and n^S (SUB)

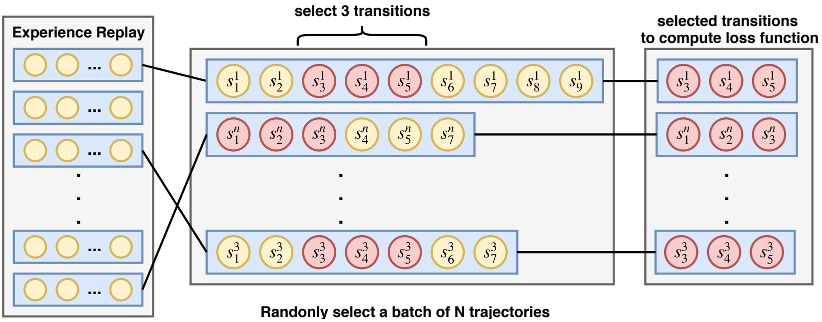


Figure: Bootstrapped Random Updates

hDRQN: Pseudo code

Algorithm 1 hDRQN in POMDP

Require:

POMDP $M = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \Omega, \mathcal{Z}\}$
Meta-controller with the network Q^M (main) and $Q^{M'}$ (target) parameterized by θ^M and $\theta^{M'}$, respectively.
Sub-controller with the network Q^S (main) and $Q^{S'}$ (target) parameterized by θ^S and $\theta^{S'}$, respectively.
Exploration rate ϵ^M for meta-controller and ϵ^S for sub-controller.
Experience replay memories M^M and M^S of meta-controller and sub-controller, respectively.
A pre-defined set of subgoals $\mathcal{G}$.
f^M and f^S are recurrent networks of meta-controller and sub-controller, respectively.

Ensure:

Initialize:
- Experiences replay memories M^M and M^S
- Randomly initialize θ^M and θ^S
- Assign value to the target networks $\theta^{M'} \leftarrow \theta^M$ and $\theta^{S'} \leftarrow \theta^S$
- $\epsilon^M \leftarrow 1.0$ and decreasing to 0.1
- $\epsilon^S \leftarrow 1.0$ and decreasing to 0.1

for $k = 1, 2, \ldots K$ **do**
  Initialize: the environment and get the start observation o
  Initialize: hidden states $h^M \leftarrow \mathbf{0}$
  while o is **not** terminal **do**
    Initialize: hidden states $h^S \leftarrow \mathbf{0}$
    Initialize: start observations $o_0 \leftarrow \hat{o}$ where $\hat{o}$ car be observation o or hidden state h^S
    Determine subgoal: $g, h^M \leftarrow$ $EPS\_GREEDY(\hat{o}, h^M, \mathcal{G}, \epsilon^M, Q^M, f^M)$
    while o is **not** terminal **and** g is **not** reached **do**
      Determine action: $a, h^S \leftarrow$ $EPS\_GREEDY(\{o, g\}, h^S, \mathcal{A}, \epsilon^S, Q^S, f^S)$
      Execute action a, receive reward r, extrinsic reward r^{ex}, intrinsic reward r^{in}, and obtain the next state s'
      Store transition $\{\{o, g\}, a, r^{in}, \{o', g'\}\}$ in M^S
      Update sub-controller $SUB\_UPDATE(M^S, Q^S, Q^{S'})$
      Update meta-controller $META\_UPDATE(M^M, Q^M, Q^{M'})$
      Transition to next observation $o \leftarrow o'$
    end while
    Store transition $\{o_0, g, r^{ex}_{total}, \hat{o}'\}$ in M^S where $\hat{o}$ can be observation o' or the last hidden state h^S
  end while
  Anneal ϵ^M and ϵ^S
end for

Experiments and Results

Experiments

- Domains:
  - Multiple goals in gridworld.
  - Multiple goals in four-rooms.
  - Montezuma's Revenge.

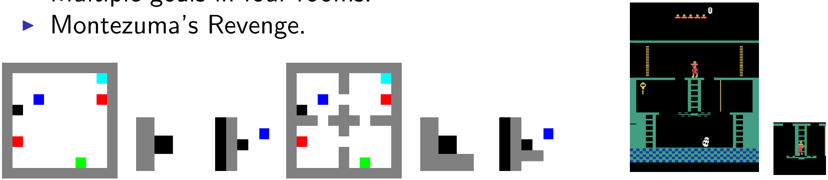


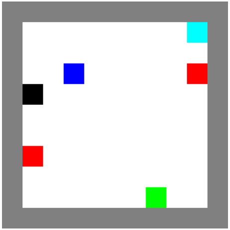
Figure: Domains

- Implementation details:
  - Tensorflow.
  - The inputs of META and SUB are a raw image of size $44 \times 44 \times 3$
  - Feature size is 256
  - Input and output of LSTM have 256 values.
  - Using ADAM to optimize the controller's parameters
  - Learning rate is 0.001
  - Discount factor is 0.99

Domain Description (1)

경희대학교
KYUNG HEE UNIVERSITY

- Multiple goal in Gridworld:
  - Gridworld map of size 11 $\times$ 11.
  - 4 types of objects: an agent (in black), two obstacles (in red) and two goals (in blue and green) or three goals (in blue, green and cyan)
  - Objects are randomly located on the map
  - Four actions: top, down, left or right.

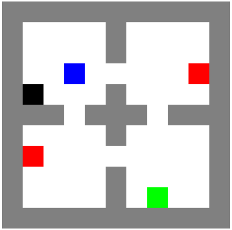


- Reward:
  - Proper order: blue $\Rightarrow$ green (two goals) or blue $\Rightarrow$ green $\Rightarrow$ cyan (three goals)
  - Classical reward: $r = \begin{cases} 1 & \text{for each reached goals in proper order} \\ -1 & \text{hit the obstacle} \end{cases}$
  - Intrinsic reward: $r^{in} = \begin{cases} 1 & \text{obtain the goal} \\ -1 & \text{hit the obstacle} \end{cases}$
  - Extrinsic reward: $r^{ex} = \begin{cases} 1 & \text{for each reached goal in proper order} \\ 0.01 & \text{otherwise} \end{cases}$

Domain Description (1)

경희대학교
KYUNG HEE UNIVERSITY

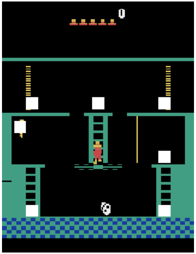
- Multiple goal in Four-rooms:
  - Four-rooms map of size 11×11.
  - 4 types of objects: an agent (in black), two obstacles (in red) and two goals (in blue and green) or three goals (in blue, green and cyan)
  - Objects are randomly located on the map
  - Four actions: top, down, left or right.



- Reward:
  - Proper order: blue $\Rightarrow$ green (two goals) or blue $\Rightarrow$ green $\Rightarrow$ cyan (three goal)
  - Classical reward: $r = \begin{cases} 1 & \text{reach goals in proper order} \\ -1 & \text{hit the obstacle} \end{cases}$
  - Intrinsic reward: $r^{in} = \begin{cases} 1 & \text{obtain the goal} \\ -1 & \text{hit the obstacle} \end{cases}$
  - Extrinsic reward: $r^{ex} = \begin{cases} 1 & \text{reach goals in order} \\ 0.01 & \text{otherwise} \end{cases}$

Domain Description (1)

- Montezuma's Revenge:
  - One of the hardest games in ATARI 2600
  - DQN achieved a score of zero
  - We use OpenAI Gym to simulate this domain
  - To pass through the doors, first, the agent needs to pick up the key.
  - Agent observes an area of 70 × 70 pixels



- Reward:
  - Classical reward: The agent will earn 100 points after it obtains the key and 300 after it reaches any door
  - Intrinsic reward:

$$r^{in} = \begin{cases} 1 & \text{reach subgoal} \\ 0 & \text{otherwise} \end{cases}$$

  - Extrinsic reward:

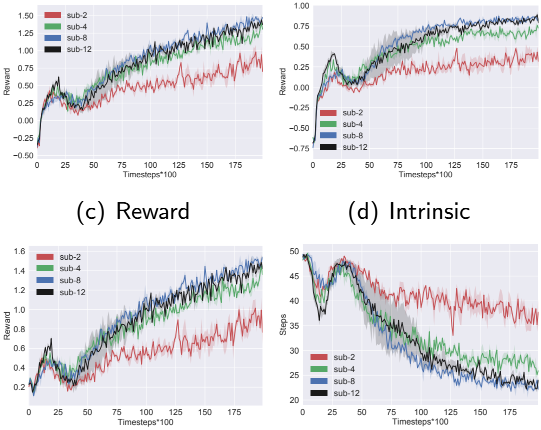
$$r^{ex} = \begin{cases} 1 & \text{obtain key or open door} \\ 0 & \text{otherwise} \end{cases}$$

Experiments

- Experiment 1: Evaluate on different values of n^M and n^S.
  - Two goals in Grid World
  - Effect of n^S
  - Effect of n^M
- Experiment 2: Evaluate on different levels of observation.
  - Two goals in Grid World
  - 3×3 observable agent
  - 5×5 observable agent
  - Fully observable agent
- Experiment 3: Compare performance of hDRQNv1, hDRQNv2 with:
  - Flat algorithms (DQN, DRQN)
  - Hierarchical algorithm (hDQN)
- Experiment 4: Montezuma's Revenge
  - Successful rate of reaching key
  - Number of times to visit the subgoals

Experiment 1: Effect of n^S (1)

- Report of hDRQNv1 with different n^S (2,4,8,12)



(c) Reward (d) Intrinsic

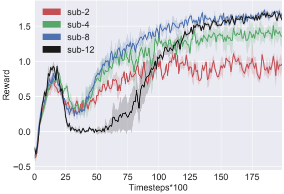
(e) Extrinsic (f) Steps

- Fixed $n^M = 1$
- Perform well with a big n^S (8,12)
- Performance decreases when n^S is decreased
- Only a little difference in performance between 8 and 12
- Intuitively, LSTM in SUB needs a long sequence of transitions

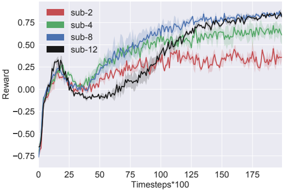
Experiment 1: Effect of n^S(2)

경희대학교 KYUNG HEE UNIVERSITY

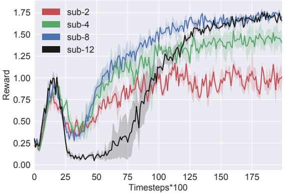
- Report of hDRQNv2 with different n^S (2,4,8,12)



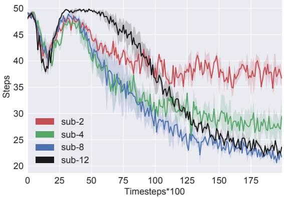
(g) Reward



(h) Intrinsic



(i) Extrinsic



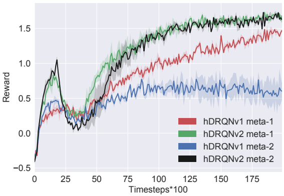
(j) Steps

- Fixed $n^M = 1$
- Same behavious as hDRQNv1

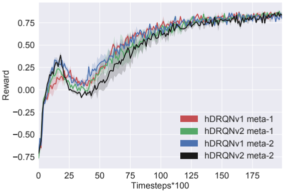
Experiment 1: Effect of n^M

경희대학교
KYUNG HEE UNIVERSITY

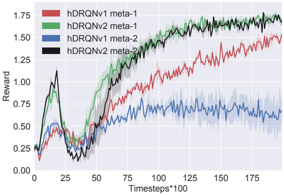
- Report of hDRQNv1 and hDRQNv2 with different n^M (1, 2)



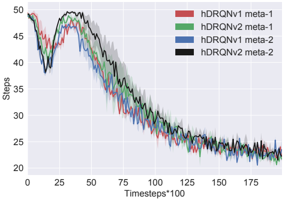
(k) Reward



(l) Intrinsic



(m) Extrinsic

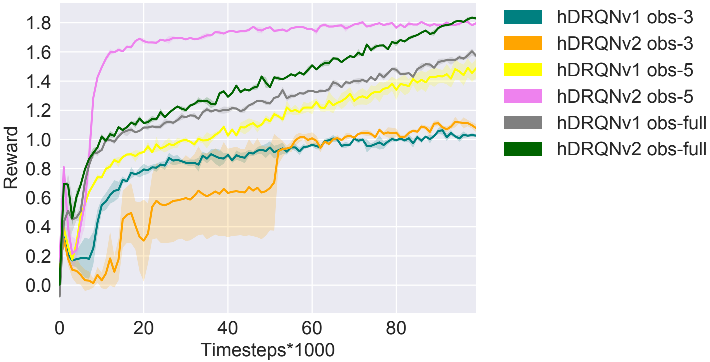


(n) Steps

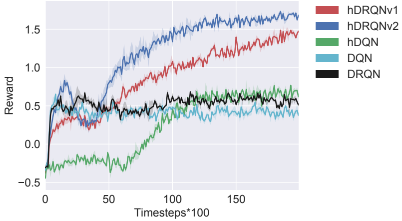
- Fixed $n^S = 8$
- With hDRQNv1, $n^M = 1$ is better than $n^M = 2$
- With hDRQNv2, the performance is the same at both settings $n^M = 1$ and $n^M = 2$

Experiment 2: Effect of different levels of observation

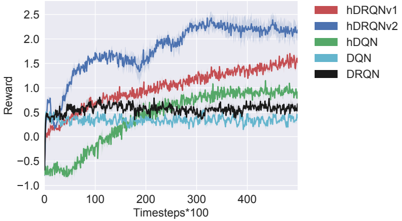
- Performance of the agent with a larger observation area is better than the agents with smaller observing abilities
- The performance of a 5×5 observable agent using hDRQNv2 seems to converge faster than a fully observable agent



- Multiple goals in gridworld
  - The hDRQN algorithms outperforms the other algorithms
  - hDRQNv2 has the best performance
  - The hDQN algorithm has poor performance in POMDP domains



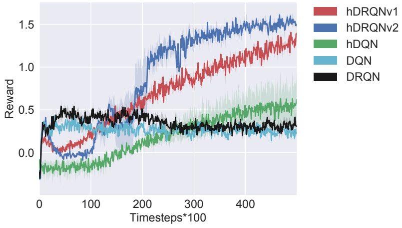
(o) Two goals in Gridworld



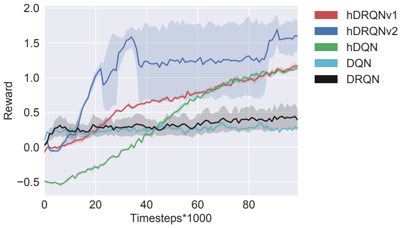
(p) Three goals in Gridworld

Experiment 3: Performance Comparison (2)

- Multiple goals in four-rooms
  - Same behavious as in Gridworld



(q) Two goals in Four-rooms

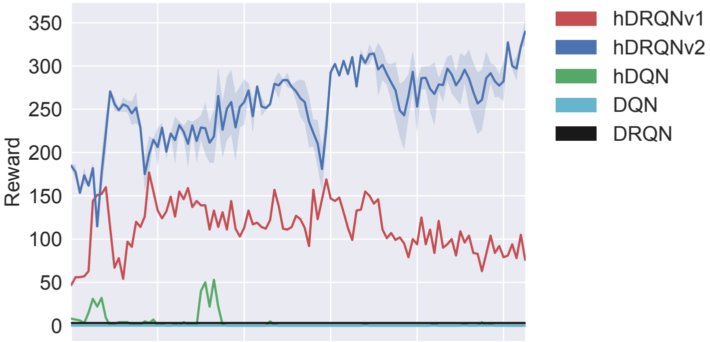


(r) Three goals in Four-rooms

Montezuma's Revenge (1)

경희대학교
KYUNG HEE UNIVERSITY

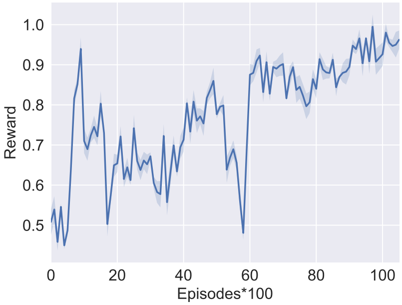
- DQN reported a score of zero
- DRQN also achieved a score of zero because of the highly hierarchical complexity of the domain
- hDQN can achieve a high score on this domain
- The hDRQNv2 algorithm shows a better performance than hDRQNv1
  $\Rightarrow$ Difference in the architecture of two frameworks has affected their performance



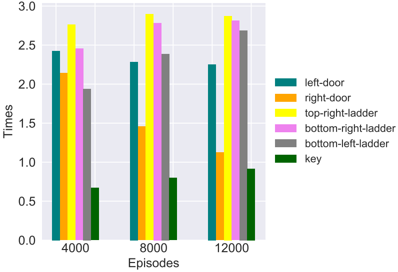
Experiment 4: Montezuma's Revenge (2)

경희대학교 KYUNG HEE UNIVERSITY

- The agent using the hDRQNv2 algorithm almost picks up the "key" at the end of the learning process
- hDRQNv2 tends to explore more often for subgoals that are on the way to reaching the "key" (E.g. top-right-ladder, bottom-right-ladder, and bottom-left-ladder)
- Exploring less often for other subgoals such as the left door and right door



(s) Success ratio



(t) Number of visits subgoals

Demo

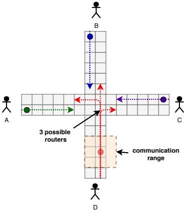
Conclusions and Future Works

Conclusions

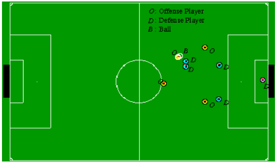
- **Implemented:** new hierarchical deep reinforcement learning algorithms (hDRQNs)
  - For hierarchical tasks
  - For both MDP and POMDP tasks
  - Takes advantage of deep neural networks (DNN, CNN, LSTM)
- **Proposed:** a new way to integrate LSTM into the learning framework, which allows to learning data efficiently and better convergence.
- **Employed:** several advanced methods in deep reinforcement learning:
  - Double Q Learning
  - Deep Recurrent Q Network
  - Dueling Q Network
  - Bootstrapped Random Updates

Future works

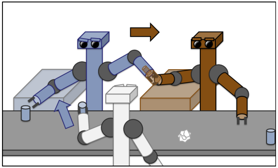
- **Improved:** our framework by tackling those problems:
  - Our framework is hard to scale for domains with more than two levels of hierarchy
  - Discovering a set of subgoals in POMDP is still a difficult problem.
- **Considered:** to apply hDRQN to multi-agent systems where the environment is partially observable and the task is hierarchical



(u) Multiple taxi co-operate to pick up and take off passengers



(v) Half Field Offense (A team of robots co-operates to score under the defense of another team)



(w) Multiple robots do a hierarchical tasks in a factory

Figure: Some hierarchical multi-agent domains

References I

경희대학교
KYUNG HEE UNIVERSITY

[1] P.-L. Bacon, J. Harb, and D. Precup, "*The option-critic architecture,*" in Proc. AAAI, 2017, pp. 1726–1734.

[2] A. S. Vezhnevets et al. (2017). "*Feudal networks for hierarchical reinforcement learning.*" [Online]. Available: https://arxiv.org/abs/1703.01161

[3] T. D. Kulkarni, K. R. Narasimhan, A. Saeedi, and J. B. Tenenbaum, "*Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation*" in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 3675–3683

[4] C.-C. Chiu and V.-W. Soo, "*Subgoal identifications in reinforcement learning: A survey,*" in Advances in Reinforcement Learning. Rijeka, Croatia: InTech, 2011.

References II

[5] M. Stolle, "*Automated discovery of options in reinforcement learning,*" Ph.D. dissertation, School Comput. Sci., McGill Univ., Montreal, QC, Canada, 2004.

[6] M. Hausknecht and P. Stone, "*Deep recurrent Q-learning for partially observable MDPs,*" in Proc. AAAI Fall Symp. Ser., 2015.

[7] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas. (2015). "*Dueling network architectures for deep reinforcement learning.*" [Online]. Available: https://arxiv.org/abs/1511.06581

[8] H. Van Hasselt, A. Guez, and D. Silver, "*Deep reinforcement learning with double Q-learning,*" in Proc. AAAI, vol. 2, 2016, p. 5

Thank You!