Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation

Presenter: Martin Braquet

November 4, 2021

Motivation

High-level description of control problem:

- Finding a policy π that maximizes expected future rewards

Major challenge in RL:

- Learning goal-directed behavior in environments with sparse feedback and delayed rewards

Difficulties:

- Insufficient exploration -> learned value function not robust

Main Problem

Limitations of prior approaches:

- Use of non-linear function approximators coupled with RL
  - learn abstractions over **high-dimensional state spaces**
  - BUT exploration with **sparse feedback** still remains a major challenge
- Boltzmann exploration and Thomson sampling
  - offer significant improvements over epsilon-greedy expiration
  - BUT limited due to the underlying models functioning at the level of basic actions

Main Problem

Importance of the problem

- Rise of complicated / high-dimensional environments
    - -> need for efficient space for exploration
- Difficult training in environments providing delayed rewards

Proposed algorithm: **Hierarchical-DQN**

- integrate **hierarchical value functions** (operating at different temporal scales)
    - top-level value function learns a policy over intrinsic goals (flexible goal specifications)
    - lower-level function learns a policy over atomic actions to satisfy the given goals
- with **intrinsically motivated deep reinforcement learning**

Problem Setting

Finding a policy π that maximizes expected future rewards

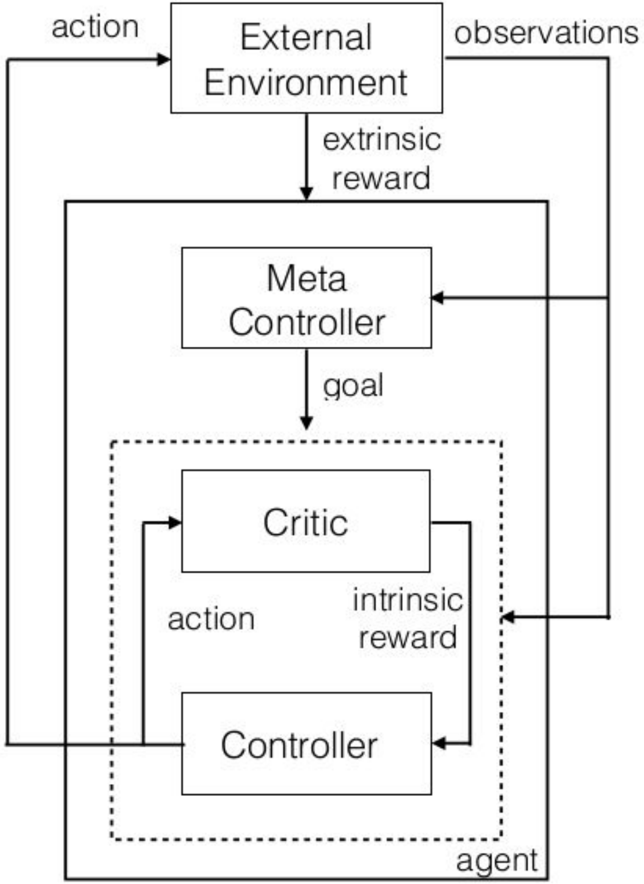
- States *s*, actions *a* and transition function T:(s,a) -> s'
- Value functions V(s, g)
  - utility of state s for achieving a given goal g
  - In high-dimensional problems, approximated by neural networks as $V(s, g; \theta)$
- (Extrinsic) reward function F(s) (to maximize over long periods of time)
- Policy π_a(s) = P(a|s)

Related Work

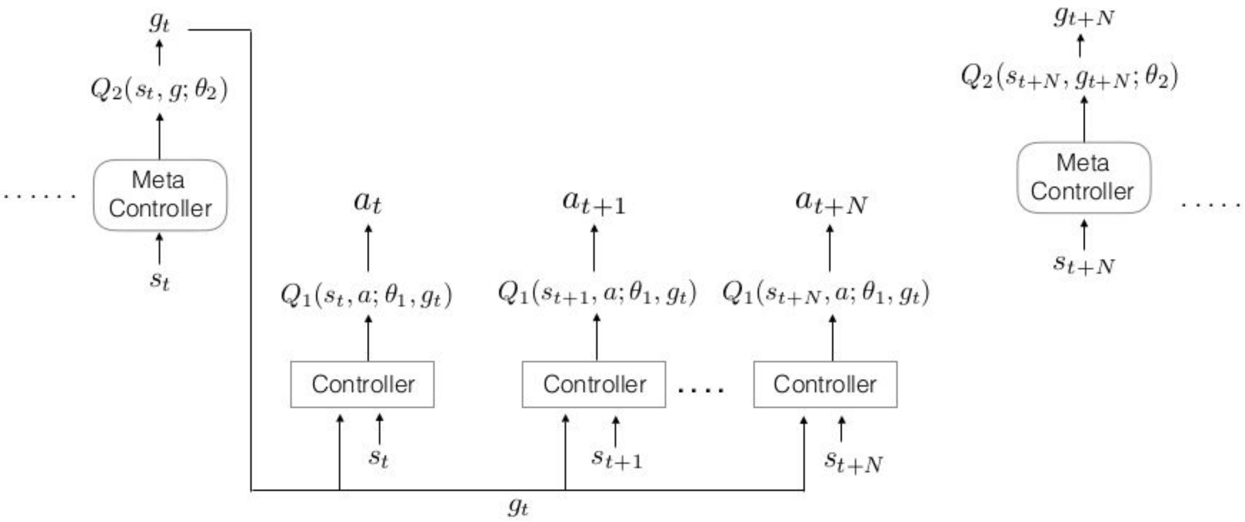
- Reinforcement Learning with Temporal Abstractions: options [1]
    - abstractions over the space of actions
    - one-step “primitive” action or a “multi-step” action policy (option)
    - generalize value functions to consider goals along with states [2] $V(s, g; \theta)$
    - Limitations: learning not shared between options, not scalable for a large number of options
- Intrinsically motivated RL
    - Design of “good” intrinsic reward functions [3]
- Deep Q-Networks
    - handle high-dimensional sensory input
    - but perform poorly on environments with sparse, delayed reward signals
    - Alleviate problem: prioritized experience replay [4] and bootstrapping [5]

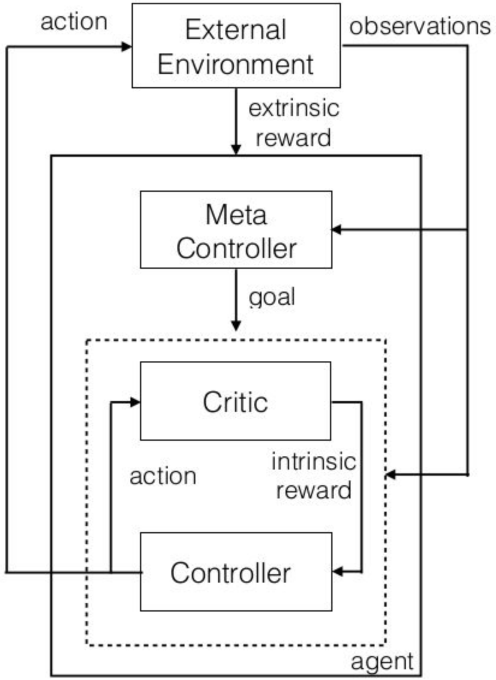
Proposed Approach

- Framework: 2 levels of hierarchy
  - top level module (**meta-controller**)
    - takes in the state and picks a new goal
  - lower-level module (**controller**)
    - uses state and goal to select actions (until goal reached or episode terminated)
- Optimize expected future intrinsic (controller) and extrinsic rewards (meta-controller)



Proposed Approach





Algorithm

- Temporal abstraction of options
  - **policies** π_g for each goal g
  - **critic**, which provides intrinsic rewards $r_t(g)$ based on whether the agent is able to achieve its goals
- Goal
  - **Controller**: maximize cumulative **intrinsic** reward $R_t(g) = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}(g)$
  - **Meta-controller**: maximize cumulative **extrinsic** reward $F_t = \sum_{t'=t}^{\infty} \gamma^{t'-t} f_{t'}$

Algorithm

Policy learning via Deep Q-Learning

- Controller

$$Q_1^*(s, a; g) = \max_{\pi_{ag}} \mathrm{E}[\sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'} \mid s_t = s, a_t = a, g_t = g, \pi_{ag}]$$

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

- Meta-controller

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

Algorithm

Represent $Q^*(s,g) \approx Q(s,g;\theta)$ as deep Q-network (same procedure for Q1 and Q2)

- Trained by minimizing loss functions

$$L_1(\theta_{1,i}) = \mathrm{E}_{(s,a,g,r,s')\sim D_1}[(y_{1,i} - Q_1(s,a;\theta_{1,i},g))^2],$$

$$y_{1,i} = r + \gamma \max_{a'} Q_1(s',a';\theta_{1,i-1},g)$$

- Update parameters θ_1 via stochastic gradient descent

$$\nabla_{\theta_{1,i}} L_1(\theta_{1,i})$$
$$= \mathrm{E}_{(s,a,r,s'\sim D_1)}\left[\left(r + \gamma \max_{a'} Q_1(s',a';\theta_{1,i-1},g) - Q_1(s,a;\theta_{1,i},g)\right)\nabla_{\theta_{1,i}} Q_1(s,a;\theta_{1,i},g)\right)\right]$$

Algorithm

Learning parameters of h-DQN

	Controller	Meta-controller
Experiences collected	At every time step	When controller terminates
Epsilon-greedy policy	$\epsilon_{1,g}$, dependent on current empirical success rate of reaching g	ϵ_2

Algorithm 1 Learning algorithm for h-DQN

Initialize experience replay memories $\{\mathcal{D}_1, \mathcal{D}_2\}$ and parameters $\{\theta_1, \theta_2\}$ for the controller and meta-controller respectively.
Initialize exploration probability $\epsilon_{1,g} = 1$ for the controller for all goals g and $\epsilon_2 = 1$ for the meta-controller.
for $i = 1, num\_episodes$ **do**
  Initialize game and get start state description s
  $g \leftarrow \text{EPSGREEDY}(s, \mathcal{G}, \epsilon_2, Q_2)$
  while s is not terminal **do**
    $F \leftarrow 0$
    $s_0 \leftarrow s$
    while not (s is terminal **or** goal g reached) **do**
      $a \leftarrow \text{EPSGREEDY}(\{s, g\}, \mathcal{A}, \epsilon_{1,g}, Q_1)$
      Execute a and obtain next state s' and extrinsic reward f from environment
      Obtain intrinsic reward $r(s, a, s')$ from internal critic
      Store transition $(\{s, g\}, a, r, \{s', g\})$ 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
    Store transition (s_0, g, F, s') 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
  Anneal ϵ_2 and adaptively anneal $\epsilon_{1,g}$ using average success rate of reaching goal g.
end for

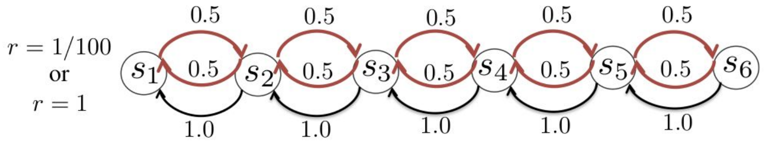
Algorithm 2 : $\text{EPSGREEDY}(x, \mathcal{B}, \epsilon, Q)$

if random() $< \epsilon$ **then**
  return random element from set $\mathcal{B}$
else
  return $\text{argmax}_{m \in \mathcal{B}} Q(x, m)$
end if

Algorithm 3 : $\text{UPDATEPARAMS}(\mathcal{L}, \mathcal{D})$

Randomly sample mini-batches from $\mathcal{D}$
Perform gradient descent on loss $\mathcal{L}(\theta)$ (cf. (3))

Experimental Setup



Stochastic decision process

- Extrinsic reward depends on the history of visited states
- 6 possible states and starts at s2
  - action left: moves left deterministically
  - action right: moves right with probability 50% (left otherwise)
- Terminal state: s1 (receives r = 100 if went through s6)

Experimental Results

Stochastic decision process

Tested against Q-learning baseline:

- which converges to sub-optimal policy of reaching state s1 directly (low reward)

h-DQN:

- learns to choose goals s4, s5 or s6
- visit s6 before going back to s1
- High reward

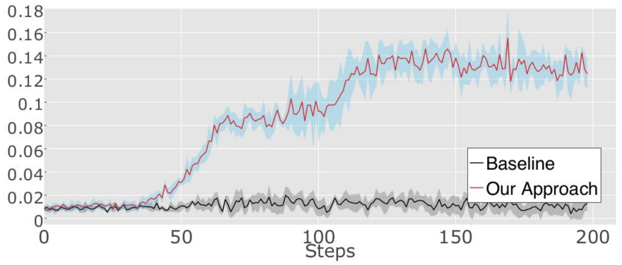


Figure 3: Average reward for 10 runs of our approach compared to Q-learning.

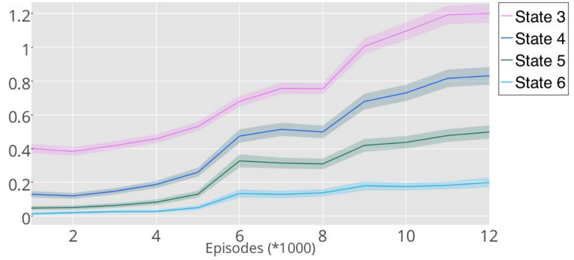


Figure 4: Number of visits (for states s_3 to s_6) averaged over 1000 episodes. The initial state is s_2 and the terminal state is s_1.

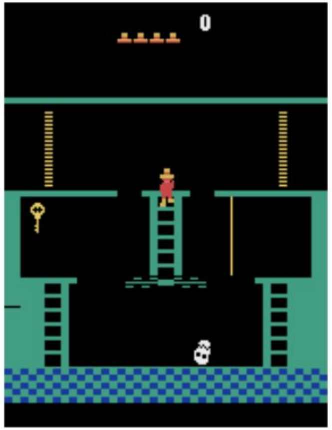
Experimental Setup

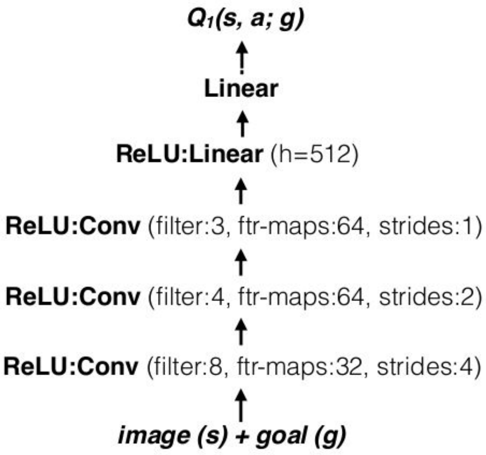
Montezuma's Revenge (ATARI game)

- Player finds a key (small reward)
- Then opens a door (high but delayed reward)

Setup

- DQN architecture for controller / meta-controller
- Goals: intermediate locations in the image





Experimental Results

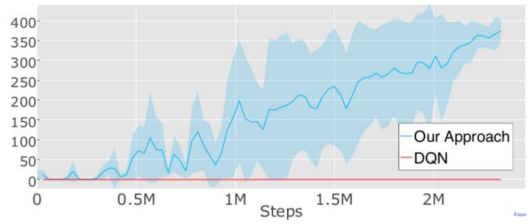
Montezuma's Revenge (ATARI game)

Tested against deep Q-learning baseline:

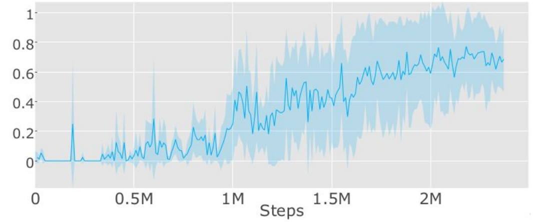
- which converges to sub-optimal policy of reaching state s1 directly (low reward)

h-DQN:

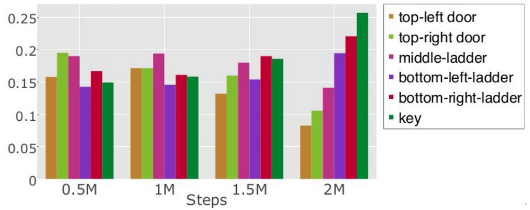
- Reach the key more often
- Obtain higher reward
- Select the appropriate intermediate goals



(a) Total extrinsic reward



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



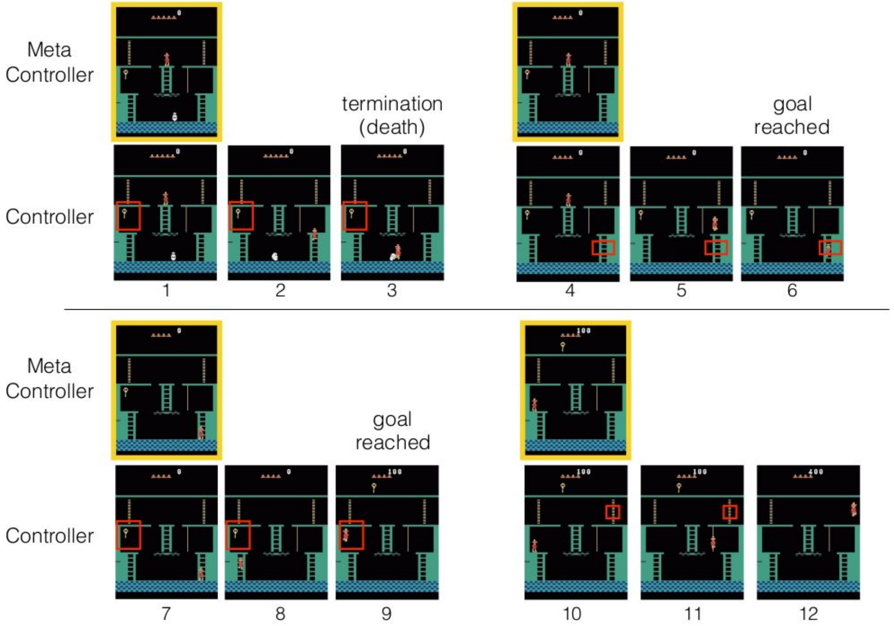
(c) Success % of different goals over time

Discussion of Results

Selecting intermediate goals

to achieve higher global reward

- Especially when rewards are delayed
- Outperform baselines Q-learning and DQN methods
- The goals are initially evenly explored
- Then most promising goals further explored



Limitations and Future Work

- Requires careful task-specific design and on-policy training
    - design of meta-controller (application dependant)
    - difficult to apply in real-world scenarios
- Missing components
    - automatically disentangling objects from raw pixels
    - agent needs to store a history of previous goals, actions and representations
- Future work
    - combination of deep generative models of images with h-DQN
    - scale up to harder non-Markovian settings
        - use of recurrent neural networks / short-term memory

Extended Readings

- Nachum, Ofir, et al. "Data-efficient hierarchical reinforcement learning." arXiv preprint arXiv:1805.08296 (2018). (https://arxiv.org/abs/1805.08296)
  - general: no onerous additional assumptions beyond standard RL algorithms
  - efficient: can be used with modest numbers of interaction samples more suitable for real-world problems, robotic control)
- Xue Bin Peng, Glen Berseth, Kangkang Yin, and Michiel Van De Panne. 2017. DeepLoco: dynamic locomotion skills using hierarchical deep reinforcement learning. ACM Trans. Graph. 36, 4, Article 41 (July 2017), 13 pages. DOI:https://doi.org/10.1145/3072959.3073602
  - two-level hierarchical control framework: robust walking gaits (low level) and motion to target (high level)
  - Simulated on a 3D biped
- Z. Yang, K. Merrick, L. Jin and H. A. Abbass, "Hierarchical Deep Reinforcement Learning for Continuous Action Control," in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 11, pp. 5174-5184, Nov. 2018, doi: 10.1109/TNNLS.2018.2805379.
  - compound skills and basic skills learned by two levels of hierarchy

References

1. R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. Artificial intelligence, 112(1):181–211, 1999.
2. T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In Proceedings of the 32nd International Conference on Machine Learning (ICML-15), pages 1312–1320, 2015.
3. S. P. Singh, A. G. Barto, and N. Chentanez. Intrinsically motivated reinforcement learning. In Advances in neural information processing systems, pages 1281–1288, 2004.
4. T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. arXiv preprint arXiv:1511.05952, 2015.
5. I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. arXiv preprint arXiv:1602.04621, 2016.

Summary

- Problem: learning goal-directed behavior in environments with sparse feedback and delayed rewards
- Use Hierarchical Deep Reinforcement Learning to motivate agents with intermediate rewards
- Prior work limited by the delay of the reward (Q-learning, DQN, etc)
- Build a two-level hierarchical model
  - operating at different temporal scales
  - top-level value function learns policy over goals
  - lower-level function learns policy over actions to satisfy goals
- Obtain higher rewards against the baselines (Q-learning) which do not succeed in obtaining the delayed rewards