Hierarchical Reinforcement Learning

Francesco Saverio Varini
Motivation: Why Hierarchical Reinforcement Learning?

- Sparse and long-term rewards!
- Not efficient long-term planning!
- Training is inefficient
- Local minima

2) The Option-Critic Architecture (Pierre-Luc Bacon, Jean Harb, Doina Precup) (September 2016)
Two papers – Two hierarchical abstractions

Hierarchical Deep Reinforcement Learning

Hierarchical abstraction over states

Hierarchical abstraction over actions
Two papers – Two hierarchical abstractions

Hierarchical Deep Reinforcement Learning

Hierarchical abstraction over states
Sparse and long-term rewards!

Solution:

Introduce *intrinsic motivation* (sub-goals + sub-rewards)
Hierarchical abstraction over states

- «Hierarchical Value functions»

**TOP LEVEL VALUE FUNCTION**

\textit{policy} over intrinsic sub-goals

**LOW LEVEL VALUE FUNCTION**

\textit{policy} over atomic actions (w.r.t the sub-goal)
Hierarchical abstraction over states

TOP LEVEL VALUE FUNCTION

WHAT NOW?

REACH THE BOTTOM
RIGHT LADDER

WHAT NOW?

REACH THE KEY

LOW LEVEL VALUE FUNCTION
Hierarchical Value Function

**Meta-Controller**
Input: state \( s \)

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

**Controller** (as standard DQN + sub-goal)
Input: state \( s \) + sub-goal \( g \)

\[
Q_1^*(s, a; g) = \max_{\pi_{ag}} \mathbb{E} \left[ r_t + \gamma \max_{a_{t+1}} Q_1^*(s_{t+1}, a_{t+1}; g) \mid s_t = s, a_t = a, g_t = g, \pi_{ag} \right]
\]
Hierarchical abstraction over states

- **Complicated** original solution space *(meta-controller)*

  "Shape" the solution space towards the specific goal *(controller)*
Example
Example
Example
Example

FARTHEST KEY!
Hierarchical abstraction over states
Hierarchical abstraction over states

- The Meta-controller (CNN + dense)
- The Controller (CNN + dense)
Hierarchical abstraction over states

- The **Internal Critic** ( <entity_1, relation, entity_2> )

![Diagram showing hierarchical abstraction over states with the Internal Critic](image)
Training

The algorithm

Controller

Meta-controller

for $i = 1, \text{num\_episodes}$ do

Initialize game and get start state description $s$

$g \leftarrow \text{EPSGREEDY}(s, G, \epsilon_2, Q_2)$

while $s$ is not terminal do

$F \leftarrow 0$

$s_0 \leftarrow s$

while not (s is terminal or goal $g$ reached) do

$a \leftarrow \text{EPSGREEDY} \{s, g\}, A, \epsilon_{1,g}, Q_1\}$

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

$g \leftarrow \text{EPSGREEDY}(s, G, \epsilon_2, Q_2)$

end if

end while

Anneal $\epsilon_2$ and adaptively anneal $\epsilon_{1,g}$ using average success rate of reaching goal $g$.16
Results: Montezuma’s Revenge

(a) Total extrinsic reward

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

(c) Success % of different goals over time
Results: Montezuma’s Revenge

Learning from Artificial Demonstrations

Rewards over sub-goals:

+ Reduce sparsity
+ Rewards are less long-term
+ Training is more efficient (?)
+ Local minima (?)
Two papers – Two hierarchical abstractions

Hierarchical Deep Reinforcement Learning

Hierarchical abstraction over states

Hierarchical abstraction over actions
Hierarchical Deep Reinforcement Learning

Hierarchical abstraction over actions

Two papers – Two hierarchical abstractions
Hierarchical abstraction over actions

- **Not efficient** long-term planning!

Solution:

- **Long-term planning** w.r.t actions (options)
Hierarchical abstraction over actions

Key points:

- The action space may be **sparse** given the current game state.

- **Strategies** to solve the current task. **Learn** them!
Hierarchical abstraction over actions
The Option-Critic architecture

- Markovian option $\omega$ in $\Omega$ is a triple $(I_\omega, \pi_\omega, \beta_\omega)$
- $I_\omega$ is an initiation set ($I_\omega \subseteq S$)
- $\Pi_\omega$ is an intra-option policy
- $\beta_\omega$ is a termination function ($\beta_\omega : S \rightarrow [0,1]$)
The Option-Critic Q-learning formula

- Option value function

\[ Q_\Omega(s, \omega) = \sum_a \pi_{\omega, \theta}(a | s) \left( r(s, a) + \gamma \sum_{s'} P(s' | s, a) \left( (1 - \beta_{\omega, \theta}(s'))Q_\Omega(s', \omega) + \beta_{\omega, \theta}(s') \max_{\bar{\omega}} Q_\Omega(s', \bar{\omega}) \right) \right) \]
Option critic architecture

- Arcade Learning Environment
Seaquest options

- Options: distributions over primitive actions

<table>
<thead>
<tr>
<th>Primitive actions</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Hierarchical abstraction over actions

- Upward shooting sequences
- Downward shooting sequences
Results - ATARI

Rainbow
428200

Microsoft AI
1 M (game completion)
https://www.youtube.com/watch?v=TpB1B9Tr_ck

Dueling DDQN
50254

A3C
24622
Hierarchical abstraction over actions

**Abstraction over action space:**

- Long-term planning
- How many options (?)
- Training is more efficient (?)
- Local minima (?)
### Summary and conclusions

- **Abstraction over states**

- **Abstraction over actions**

#### Table Comparison

<table>
<thead>
<tr>
<th>1° paper</th>
<th>2° paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adopted temporal abstraction (Sutton et al.)</td>
<td>Adopted temporal abstraction (Sutton et al.)</td>
</tr>
<tr>
<td>Sub-goals <em>(meta-controller)</em></td>
<td>Options</td>
</tr>
<tr>
<td>Transfer learning (among sub-goals)</td>
<td>Transfer learning (among options)</td>
</tr>
<tr>
<td>Specify sub-goals. Establish extra rewards</td>
<td>Specify the <strong>number</strong> of options. Also, no extra rewards.</td>
</tr>
<tr>
<td>Scalable to multiple sub-goals <em>(same DQN controller)</em></td>
<td>Scalable to multiple options <em>(same DQN)</em></td>
</tr>
</tbody>
</table>
Summary and conclusions

- Abstraction over states

Abstraction over actions

NO MEMORY W.R.T THE HIERARCHICAL ABSTRACTION (sub-goals/options)  
→ Next talk

1° paper
Adopted temporal abstraction (Sutton et al.)

<table>
<thead>
<tr>
<th>Sub-goals (meta-controller)</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer learning (among sub-goals)</td>
<td>Transfer learning (among options)</td>
</tr>
<tr>
<td>Specify sub-goals. Establish extra rewards</td>
<td>Specify the number of options. Also, no extra rewards.</td>
</tr>
<tr>
<td>Scalable to multiple sub-goals (same DQN controller)</td>
<td>Scalable to multiple options (same DQN)</td>
</tr>
</tbody>
</table>
End

- Thanks!

QUESTIONS?
Hierarchical abstraction over states
Hierarchical abstraction over states – Test screenshot
Hierarchical abstraction over states – Test screenshots
Hierarchical abstraction over states – Test screenshots
Hierarchical abstraction over states – Test screenshots
Hierarchical abstraction over states – Loss functions

- There are two loss functions associated to the controller and meta-controller respectively.

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

- Its gradient is

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

- Similarly the gradient for \( L_2 \) can be derived
Hierarchical abstraction over actions
Hierarchical abstraction over actions

- Option-critic with tabular intra-option Q-learning
Hierarchical abstraction over actions

- Advantage function + Regularizer

“As a consequence of optimizing for the return, the termination gradient tends to shrink options over time. This is expected since in theory primitive actions are sufficient for solving any MDP.”

\[ A_\Omega(s, \omega) + \xi = Q_\Omega(s, \omega) - V_\Omega(s) + \xi \]

“This makes the advantage function positive if the value of an option is near the optimal one, thereby stretching it.”